ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

DESARROLLO DE UNA INTERFAZ GRÁFICA CON PANTALLA TÁCTIL PARA EL MANEJO DE UN SISTEMA ROBÓTICO

Alumno: Javier Echeverria Senosiain
Tutor: Jesús María Corres Sanz
Pamplona, 18 de Noviembre de 2011

ÍNDICE

0. PRESENTACIÓN Y AGRADECIMIENTOS

El trabajo que se expone a continuación plasma de una manera general el proyecto fin de carrera presentado por el alumno Javier Echeverria Senosiain bajo la tutela del profesor Dr. Jesús María Corres Sanz para el departamento de Ingeniería Eléctrica y Electrónica de la Universidad Pública de Navarra.

Este proyecto surge con la idea de sustituir un ordenador de sobremesa que controla un sistema robótico por un equipo de menor tamaño, portátil y manejable.

Para la realización de este proyecto era necesario tener conocimientos de programación, conocimientos de microcontroladores y de la transmisión de datos.

Tras la elección del sistema operativo y el lenguaje de programación, se pudo observar que en estos dispositivos, las librerías que son administradas para programar son bastante menores a las proporcionadas para un equipo de sobremesa, por lo que hubo varios problemas en la realización de la conexión entre los dispositivos.

Con esto dimos por finalizado el proyecto, aunque como en todo trabajo, todo es ampliable y mejorable.

Agradecimientos

A mis padres y hermano, a los cuales han pasado a mi lado los buenos y duros momentos de mi trayectoria educativa.

A Maitane, Iñigo e Iker por estar a mi lado estos años tan fabulosos en la universidad.

A Cristian y Fernando por hacerme pasar un año lejos del hogar como si estuviese en mi propia casa.

A todos los integrantes de "Bajera Viajera" por pasar todos los días cerca de mí.

Al profesor Dr. Jesús María Corres Sanz por su ayuda en este proyecto.

Y a todos los que no he nombrado, que deberían ser nombrados

A todos, muchas gracias.

1. INTRODUCCIÓN

1.1 Objetivos del proyecto.

El objetivo de este proyecto fin de carrera es la creación de una aplicación gráfica para una pantalla táctil. En esta aplicación se introducirán los datos y controles necesarios para el manejo de un brazo robótico con nanotecnología.

1.2 Desarrollo del proyecto.

El primer paso fue estudiar el programa (visto en el punto 1.3) del cual se debía partir. Tras estudiar las diferentes opciones que se podían adoptar con los instrumentos de los que se disponían, se decidió el sistema operativo y el lenguaje de programación que más adelante se utilizó.

El segundo paso fue la configuración y compilación del sistema operativo. Al ser un hardware con limitaciones tanto en procesamiento y memoria, se decidió instalar las opciones básicas para el funcionamiento del sistema operativo.

Después de tener una imagen creada, el cuarto paso fue la instalación y configuración de dicho sistema para que reconociera todos los dispositivos que se disponen a nivel hardware.

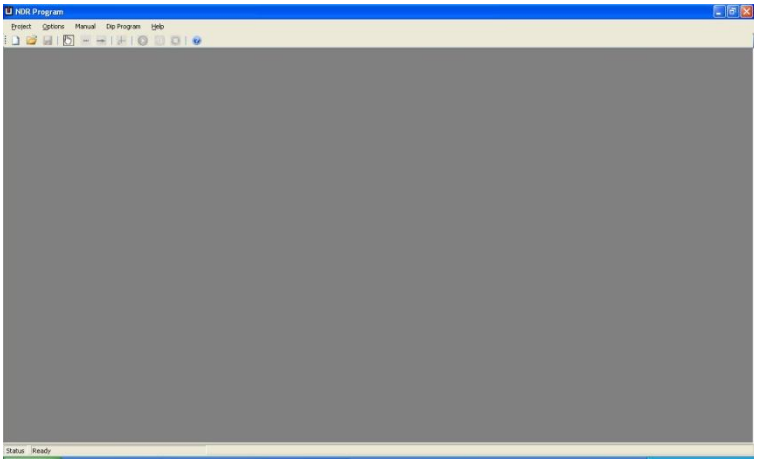
Una vez instalado, el quinto paso fue la realización de una interfaz gráfica sencilla para la introducir los tres datos necesarios que se transmiten al “PIC DEM FS USB”.

Tras la comprobación de una transmisión satisfactoria entre el dispositivo “Friendly ARM” y el “PIC DEM FS USB”, el sexto paso fue la creación de una interfaz gráfica para introducir los datos que realmente eran necesario, ya que los datos introducidos en el anterior pasos son calculados a través de estos nuevos.

1.3. Datos de partida

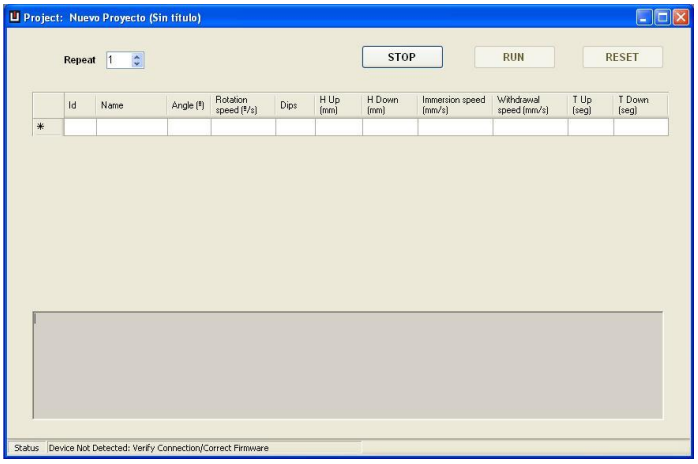
En esta sección, describiremos el programa "NDR Program", en el que se basará el programa que se va a crear en este proyecto.

Se puede observar en la siguiente imagen el menú principal del programa como tenemos las opciones básicas de nuevo, abrir, guardar y manual. Dependiendo de la opción deseada tendremos otras ventanas.

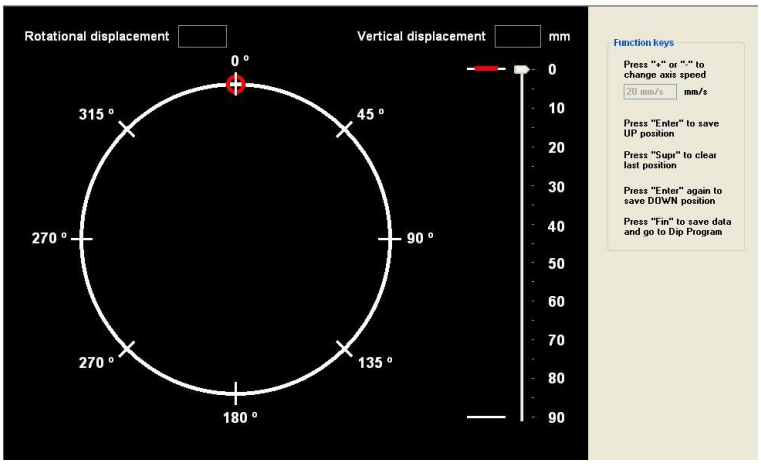


Si pulsamos en nuevo o en abrir, obtendremos una nueva ventana en la cual podremos:

- Indicar el número de repeticiones que se va a ejecutar.
- Iniciar, parar y resetear el brazo mecánico mediante los botones "Stop", "Run" y "Reset".
- Introducir, modificar o eliminar los datos de la tabla.
- Leer la situación o estado en el que se encuentra el programa mediante el recuadro de texto.
- Conocer la posición del brazo, en una segunda barra de estado que en la siguiente imagen no se puede observar.
- Conocer el estado del brazo en la barra de estado.



En cambio, si pulsamos el botón de manual, se abrirá una nueva ventana con la cual, mediante el teclado, podemos manejar el robot según pulsemos las teclas.



Una vez revisado el programa a nivel de usuario, vamos a observar el nivel de programador. Este programa está escrito en el lenguaje de programación C# con las librerías ".NET Framework 4.0". Para ello, se ha utilizado el programa "Microsoft Visual Studio 2010" para poder escribirlo y compilarlo.

Hay otra parte del proyecto que se necesitó una pequeña revisión. La placa "PIC DEM FS USB" necesitó ser programada. Por ello, tras un ver un poco el código, se puede observar que está escrito en el lenguaje de programación C y para poder escribir, compilar y grabar se utiliza el programa "MPLAB IDE".

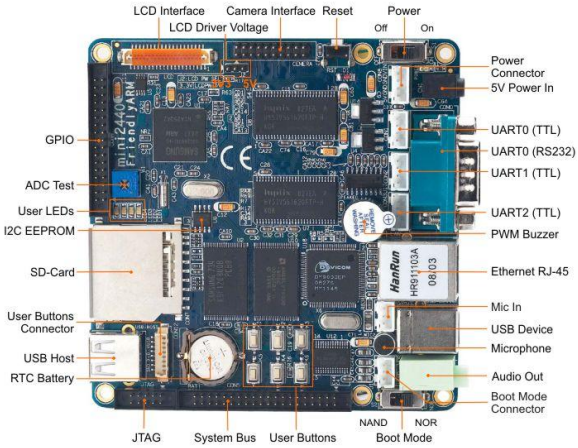
2. DESCRIPCIÓN DEL HARDWARE

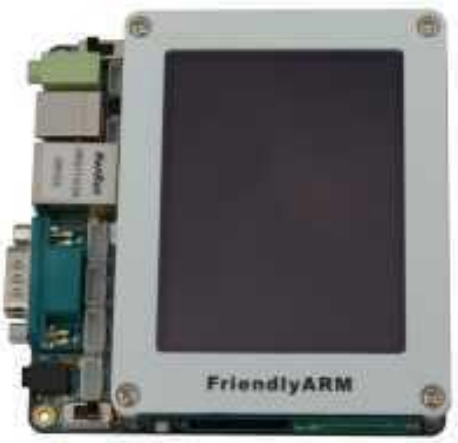
2.1. Friendly ARM Mini2440

Este es nuestro primer dispositivo y dispone de:

- 64MB de memoria RAM
- 128MB de memoria NAND para el sistema operativo
- 2MB de memoria NOR para la BIOS
- Conector USB Host y USB Device
- Conector RS232
- Conector para pantalla táctil de 7 (800x480 pixeles) o 3.5 (240x320 pixeles) pulgadas.

Entre este y el siguiente dispositivo se creara una conexión física para la transmisión de datos.



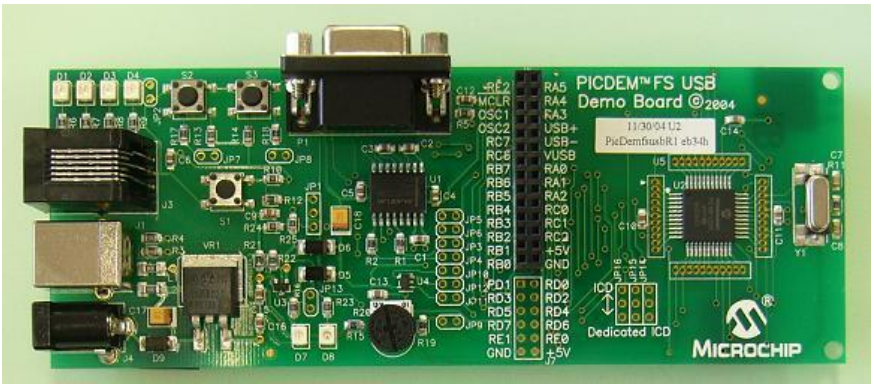


2.2. PIC DEM FS USB

Este es nuestro segundo dispositivo y dispone de:

- Conector USB Host
- Conector RS232
- PIC 18F4550

Este dispositivo tiene más características, pero estas son las que más nos interesan para nuestro proyecto. Este recibirá ordenes del “Friendly ARM Mini 2440” y enviara pulsos eléctricos al “Microstepping Driver KL 4020”.



2.3. Microstepping Driver KL 4020

Este dispositivo permite contar los pulsos generados por el "PIC DEM FS USB" y cada cierto número de pulsos, genera un pulso para la realización del movimiento del siguiente dispositivo.

2.4. LR43KH4R-05-940ENG 5VDC

Este dispone de dos motores, uno para girar y el otro para el desplazamiento vertical.

3. SISTEMA OPERATIVO PARA EL FRIENDLY ARM

Según la página web del dispositivo, se puede observar que tiene una gran posibilidad de configuraciones. Se puede elegir varios sistemas operativos, entre los cuales se encuentran Android, Linux y Windows CE.

Para las versiones de Android y Linux se puede descargar las imágenes del sistema operativo. En cambio, para el sistema operativo Windows CE, se puede descargar un fichero de configuración para adaptarlo al dispositivo, con múltiples opciones a elegir.

La razón de elegir "Windows CE 6.0" es por si deseamos modificar el código del producto, ya que está escrito en C#, y ese lenguaje es compatible solo con Windows.

Para crear una imagen del sistema operativo "Windows CE 6.0" se necesitan varios programas:

- Microsoft Visual Studio 2005
- Windows CE 6.0 y sus actualizaciones "Platform builder SP1", "R2" y "R3"
- El SDK del dispositivo para Windows CE 6.0

3.1. Elección de las características del fichero de configuración para el Windows CE 6.0

El fichero de configuración para Windows CE 6.0, que está presente en la página oficial del dispositivo, tiene algunos errores y no permite la compilación correcta del sistema operativo. Por ello, los usuarios que utilizan este dispositivo han creado otra versión que permite la compilación del sistema operativo. Esta se puede descargar de la dirección que se encuentra en el anexo 1, dirección 1 y pulsando en "Mini2440BSP.zip" o desde el enlace que se encuentra en el anexo 1, dirección 2.

La versión creada por los usuarios contiene un instalador, en el que hay que elegir el modelo de pantalla y pulsar en instalar. El modelo de pantalla que utilizamos en este caso es la pantalla T35. Por defecto la configuración se instalará en la carpeta "\WINCE600".

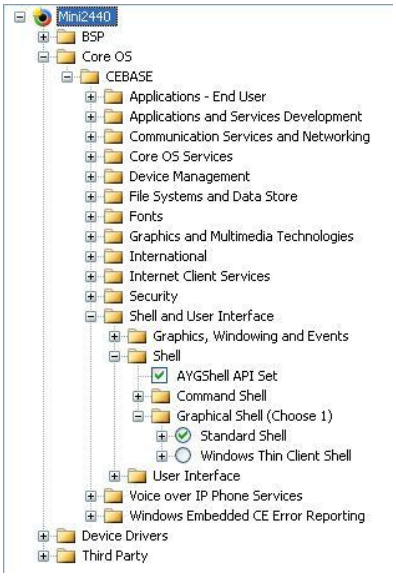
El fichero de configuración contiene, además de los drivers necesarios para el manejo del hardware, como por ejemplo el lector de tarjetas, contiene una nueva utilidad para calibrar la pantalla táctil al iniciar Windows por primera vez.

Antes de compilar este fichero, se han cambiado dos opciones. La primera opción, como se puede observar en la siguiente imagen, se encuentra en "/Core OS/Applications and Sercices Development/.NET Compact Framework 3.5/". Por defecto, esta seleccionada la versión 2.0 de estas librerías, la cual hay que cambiar por la versión 3.5. No importa mucho si no se selecciona ahora, ya que después de terminar la instalación de Windows CE, se puede instalar esta versión a través de un ordenador de sobremesa.

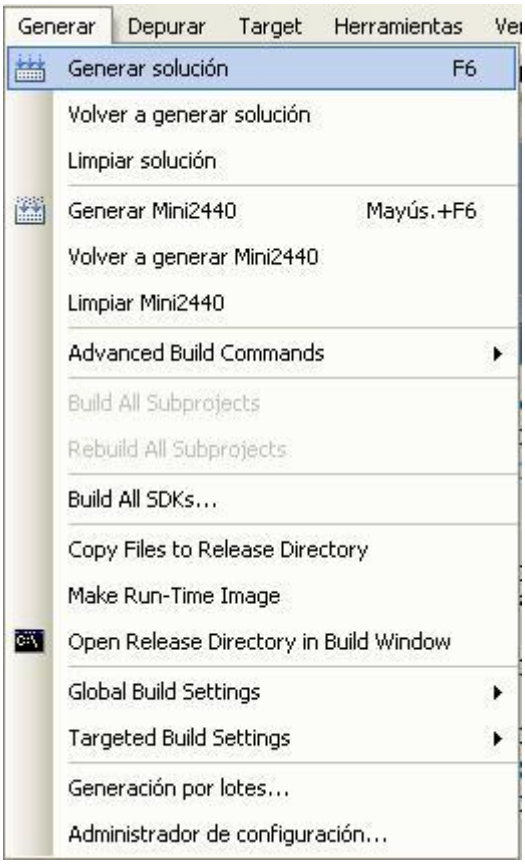
La segunda opción, como se puede observar en la siguiente imagen, y la más importante para este proyecto, se encuentra en "/Core OS/Shell and User Interface/Graphical Shell/" y solo se puede seleccionar una de las dos configuraciones que permite y son las siguientes:

- La primera es "Standar Shell" y permite tener a Windows una interfaz gráfica con la barra de inicio, mi dispositivo, papelera de reciclaje, etc.
- La segunda es "Windows Thin Client Shell" y elimina toda esa interfaz a la que estamos acostumbrados a ver respecto a un ordenador de sobremesa. Con esta no he podido trabajar ya que la pantalla principal que aparece tras instalar Windows CE en el dispositivo, es superior al tamaño de la pantalla que se utiliza en este proyecto.

En un principio se quería trabajar con la segunda opción para que el usuario final no pudiese tocar ningún archivo de Windows y poder utilizar el dispositivo con otras finalidades. En cambio, se ha buscado otra solución que más adelante, con la creación de la interfaz gráfica, se explicara.



Una vez seleccionada la configuración que se dese, iremos al menú "Generar" y pulsaremos en "Generar solución". Después de unos 10 minutos obtendremos la imagen personalizada de este sistema operativo en la carpeta "\WINCE600\OSDesigns\proyecto\Mini2440\RelDir\Mini2440_ARMV4I_Release". Una vez en esta carpeta buscaremos el archivo "NK.bin", el cual contiene el sistema operativo.

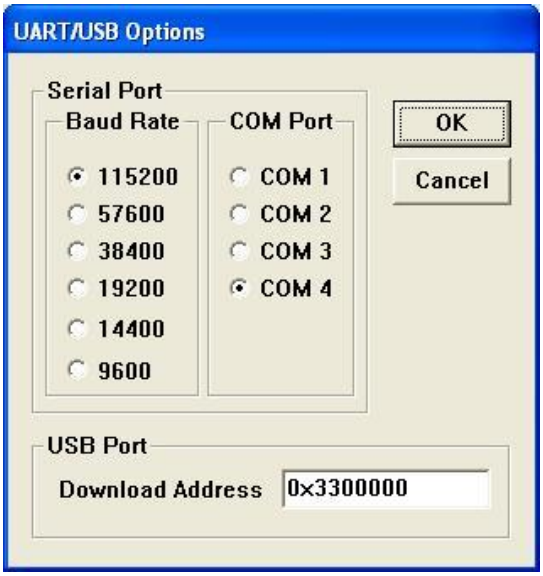


3.2. Instalación del sistema operativo desde Windows

Para realizar este paso es necesario, un ordenador y descargarse de la sección "Donwloads" de la página oficial del dispositivo, anexo 1 dirección 3, los siguientes archivos:

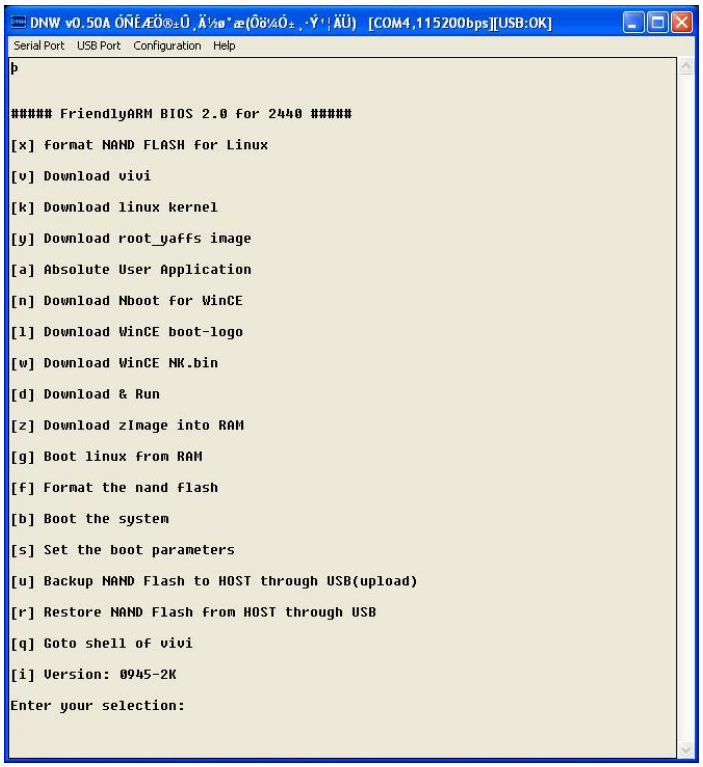
- DNW para Windows. Es un terminal para una conexión serie entre el dispositivo Friendly y el ordenador, que además permite una conexión USB para enviar los archivos necesarios.
- USB Download driver setup. Son los drivers necesarios para que el ordenador pueda detectar que se ha conectado el Friendly ARM en modo BIOS al ordenador mediante USB.
- WinCE BSP. Conjunto de archivos, en el que se encuentran los sistemas de arranques y una copia del fichero de configuración para compilar.

El primer paso es la configuración del puerto serie por el cual nos comunicaremos con el Friendly ARM. Para ello abriremos el programa "DNW.exe" e iremos al menú "Configuratión\Options". En este configuraremos la velocidad de transmisión y el puerto por el que nos comunicaremos. En nuestro caso, como se puede observar en la imagen, seleccionaremos el "COM4" y a una velocidad de 115200 baudios. Una vez configurado el puerto serie, debemos indicar cuál es la primera zona de memoria donde instalaremos el sistema operativo, que será la dirección "0x3300000".



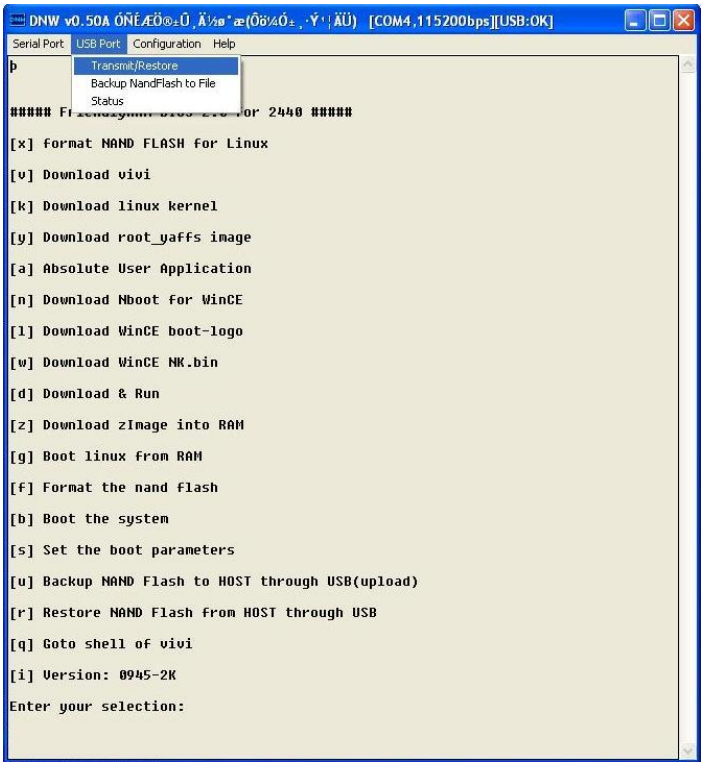
Una vez configurado el programa, deberemos habilitar la conexión serie mediante el menú "Serial Port\Connect. Si la conexión se ha realizado correctamente, en la cabecera del programa aparecerá el puerto y la velocidad de transmisión seleccionada en vez de una "X".

Antes de encender el dispositivo, deberemos cambiar el interruptor “Boot Mode” a modo “NOR”. Una vez hecho este cambio en el modo de arranque, encenderemos el dispositivo con el cable serie ya conectado entre el ordenador antes de encenderlo para poder ver el menú que manda el dispositivo. Una vez encendido, conectaremos el dispositivo al ordenador mediante USB para mandar los ficheros necesarios. Al conectarlo, si hemos instalado los drivers, la cabecera del programa “DNW” cambiara y en la sección de USB aparecerá “Ok”. Hecho esto, tendremos la siguiente imagen.



Lo que primero deberemos hacer es formatear la unidad nand con la opción “F”. La memoria nand es donde se encuentra el sistema operativo.

Una vez terminado el formateo, instalaremos el sistema de arranque. Para ello introduciremos la opción “N”. A continuación nos pedirá que enviemos el archivo que contiene el sistema de arranque. Para ello pulsaremos sobre el botón “USB Port\Transmite” y buscaremos el archivo “nboot_T35.bin” que se encuentra en la carpeta “nboot” del conjunto de archivos “WIN CE BSP”, descargado de la página oficial.



Terminado el envió del sistema de arranque, indicaremos la opción “W” para realizar la instalación de Windows sobre el dispositivo. Seguimos los pasos anteriormente descritos para enviar el archivo “NK.bin”. Este archivo, como anteriormente se ha comentado, se encuentra en “\WINCE600\OSDesigns\proyecto\Mini2440\RelDir\Mini2440_ARMV4I_Release”. Una vez terminada la instalación, el sistema operativo del dispositivo, se iniciara automáticamente.

El siguiente paso para la instalación será calibrar la pantalla, que como he dicho antes, se iniciará automáticamente con el primer arranque. Luego tendremos que ejecutar la aplicación que se encuentra en “MyDevice\Windows\” llamada “Rotate” para girar la pantalla 90 grados y asi tenerla en horizontal.

Para concluir con este paso, solo hay que volver a cambiar el interruptor “Boot Mode” a la posición “Nand”.

4. LA APLICACIÓN

4.1. Elección del lenguaje de programación

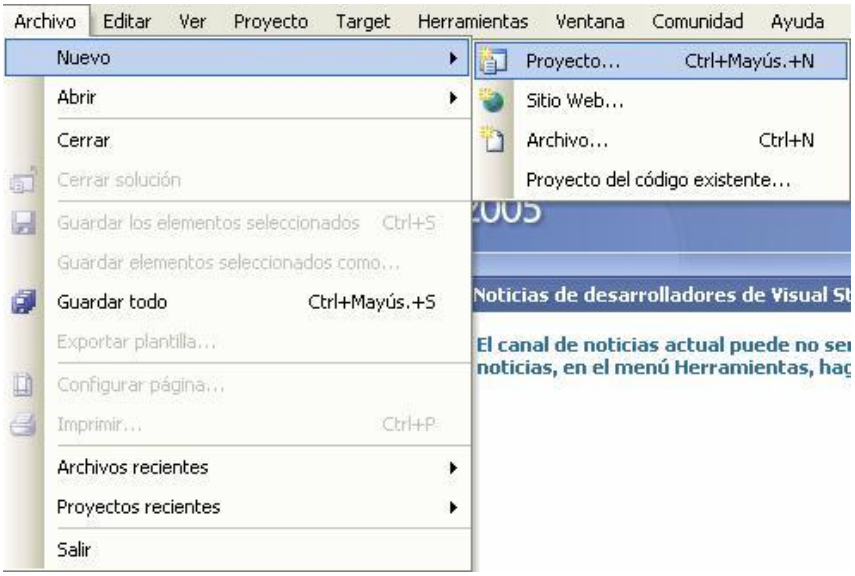
La elección del lenguaje C# para la programación de la aplicación, ha sido porque el programa original está escrito en el mismo lenguaje, y de esta forma, se puede reutilizar parte del código con pequeñas variaciones.

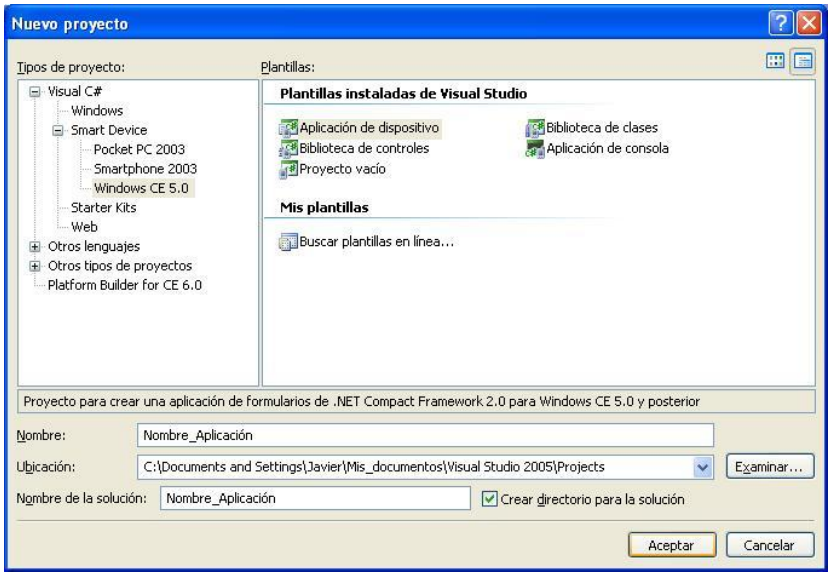
Según Microsoft, la versión utilizada de las librerías solo representa un 30% de la versión .Net Framework 4.0, que es la utilizada en el programa original, entonces, habrá que hacer ciertos cambios ya que habrá métodos, clases e incluso ficheros enteros que no estén disponibles para la versión 3.5 Compact. Estas diferencias se pueden observar mediante la dirección web número 4 que se encuentra en el anexo 1.

En el Microsoft Visual Studio 2005 esta la versión 2.0 del .Net Compact Framework. Para obtener la versión 3.5 de este producto, se puede descargar gratuitamente desde la página que se encuentra en el anexo 1 número 5. Cuando instalamos este producto también nos da la opción de instalar las librerías en un dispositivo portátil.

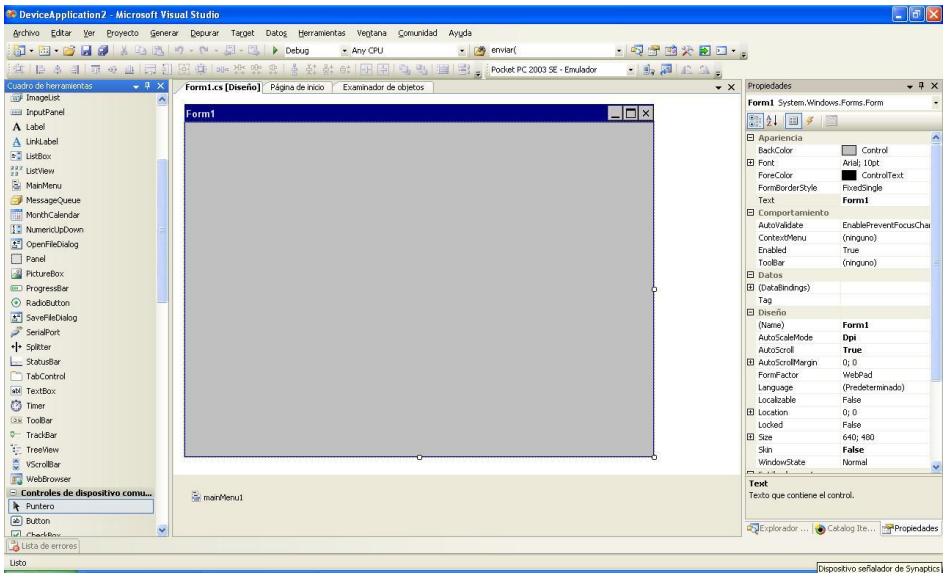
4.2. La interfaz gráfica

Para comenzar con la interfaz gráfica crearemos un nuevo proyecto con el "Microsoft Visual Studio 2005". Para ello pulsaremos en el menú "Archivo\Nuevo\Proyecto…\" y en la ventana emergente pulsamos sobre "Visual C#\Smart Device\Windows CE 5.0". Aquí podemos seleccionar desde una aplicación de consola, hasta crear una interfaz gráfica. Introducimos el nombre del proyecto y pulsamos en "Aceptar". Ambos pasos se pueden observar en la siguiente imagen.





Una vez creado el proyecto obtendremos, en nuestro caso la siguiente imagen. En ella se puede ver el editor que vamos a utilizar con las diferentes pestañas que vamos a utilizar posteriormente. En la parte de la izquierda encontraremos la pestaña “cuadro de herramientas” y en a la derecha, encontraremos “propiedades” y “explorador de soluciones”.



Para ver las librerías utilizadas en el nuevo proyecto nos dirigimos al "Explorador de soluciones, y en él, buscamos la carpeta "References", en la que aparecerán las librerías utilizadas.

Para actualizar las librerías de la versión 2.0 a la 3.5 pulsaremos con el botón secundario del ratón sobre "References" y cuando aparezca el menú, se pulsará en "Agregar Referencia". Aparecerá una nueva ventana en la que deberemos pulsar sobre "Examinar" y buscaremos las librerías necesaias en la dirección "\Archivos de programa\ Microsoft .NET\ SDK \CompactFramework\ v3.5\ WindowsCE", en la que aparecerán todas las librerías que se pueden añadir.

Se puede cambiar el nombre final de la aplicación y agregar un icono al compilado, para ello pulsaremos sobre el menú "proyecto\propiedades". En la nueva ventana, en la pestaña "aplicación" veremos el atributo "nombre del ensamblado" e "icono". Si modificamos estos dos atributos se modificará el archivo final resultante.

Nada mas crear un nuevo proyecto con una interfaz gráfica, se establecen tres objetos que, en nuestro caso, no son necesarios:

- Los botones minimizar, maximizar y cerrar
- La barra de título
- Un menú de archivo, edición, etc…

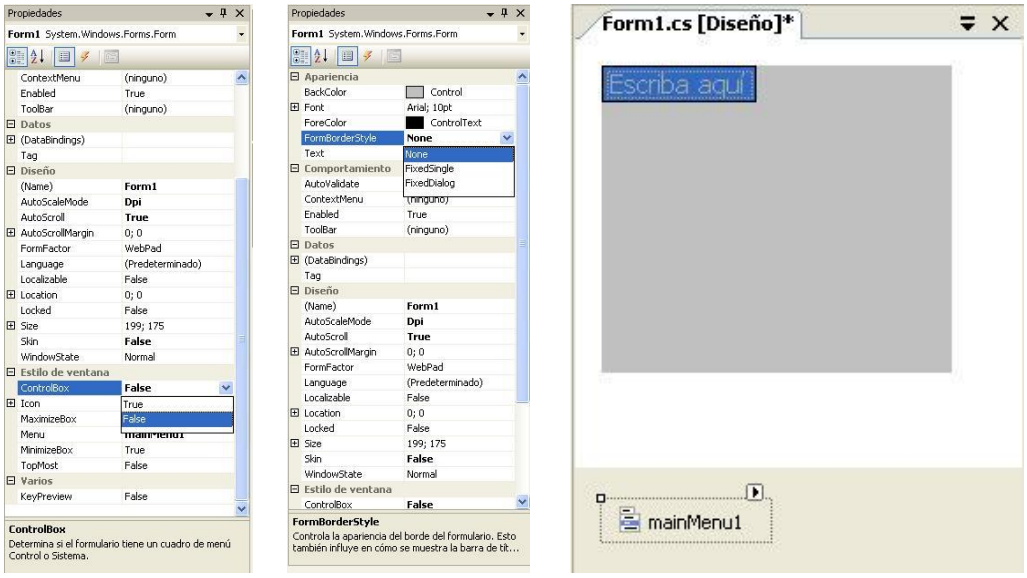
Es necesario que estos tres elementos sean eliminados, primero, para ganar espacio en la interfaz gráfica y segundo, para evitar que el usuario pueda mover la aplicación gráfica y utilizar el dispositivo para otros fines.

Para eliminar los botones, seleccionamos el formulario y en la pestaña de "propiedades", buscaremos el atributo "ControlBox" y seleccionaremos el valor "False".

Para eliminar la barra del título, en la misma pestaña, buscaremos el atributo "FormBorderStyle" y seleccionaremos el valor "None".

Por último, para eliminar el menú principal, solo hay que seleccionar el "mainMenu1" y pulsar la tecla "Supr" del teclado.

Las siguientes tres imágenes representan, de izquierda a derecha, los tres objetos que hay que eliminar respectivamente.



Una vez eliminados estos objetos, como nuestra pantalla es muy pequeña, se decidió trabajar por pestañas.

En la primera pestaña se representaría los datos introducidos mediante una tabla, junto con tres botones para abrir, guardar y vaciar el proyecto. Además de estos botones, debe aparecer el nombre del archivo e indicar el número de repeticiones que se ejecutarán los datos de la tabla.

Para poder ver todos los valores de una fila, además de realizar acciones como añadir, eliminar o modificar los valores de la tabla, se decidió crear otra pestaña en la que se dispondrá de una calculadora para introducir los datos correspondientes, y varios botones como añadir, modificar o eliminar la fila correspondiente.

Como todavía quedan datos que mostrar, se decide crear otra pestaña en la cual se mostrará los datos que faltan, además de los botones para iniciar, parar o resetear el brazo robótico.

Terminada la explicación de las pestañas, nuestra interfaz gráfica solo se compondrá de objetos como “TextBox”, “Button”, “Timer”, “TabControl”, “Label” o “LinkLabel”, “DataGrid”, “NumericUpDown”… Para añadir cualquier objeto, arrastramos el objeto deseado desde la pestaña “cuadro de herramientas”, sobre la interfaz.

La mayoría de los objetos comparten variables, como por ejemplo, la variable “Text”. Esta se puede modificar desde la pestaña “propiedades”, o si se desea modificar mientras se ejecuta el programa, hay que realizar una asignación a esta variable.

Como se puede observar en la siguiente imagen, en el primer método la variable “Text” del objeto “archivo” es sustituida por el valor de “sms”, y en el segundo método la variable “Text” es leida y devuelta al método que requiere sus servicios.

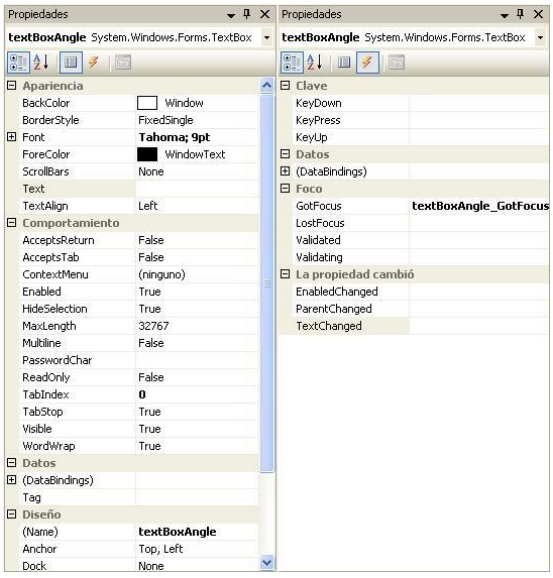
```
//Pone el nombre del archivo
public void putArchivo(string sms) {
    archivo.Text = sms;
}

//Recupera el nombre del archivo
public string getArchivo() {
    return archivo.Text;
}
```

De esta manera se pueden modificar las variables que componen a todos los objetos. Hay que tener en cuenta que las variables solo pueden ser modificadas por otra variable del mismo tipo. Por ejemplo, en la siguiente imagen, si queremos modificar la variable "Text" con un dato del tipo "Int", hay que realizar primero una conversión a tipo "String".

```
private void putLabelRow (int i){
    labelRow.Text = "Row: " + i.ToString();
}
```

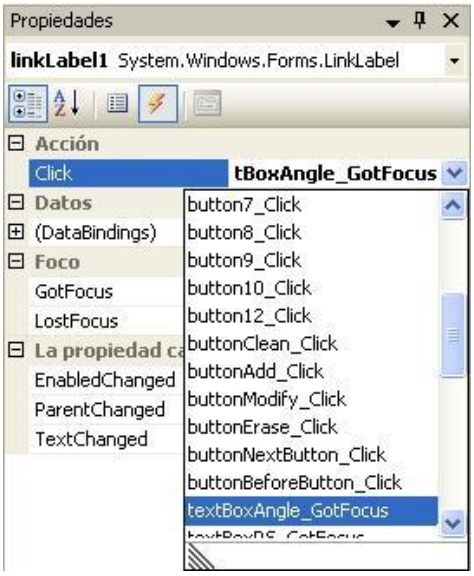
Además de las variables, los objetos también tienen "eventos". Para observarlos, en la pestaña "propiedades" pulsamos sobre "eventos" y veremos los eventos permitidos para el objeto seleccionado. Como se puede observar a la izquierda de la imagen, vemos los atributos que componen el objeto "TextBox", y en la parte de la derecha vemos los eventos de este objeto. Para añadir un evento al objeto, llevamos el cursor sobre el evento, hacemos doble click sobre él y el programa creará el código necesario, para que, en este caso, obtener el foco, salte al método programado.



El programa creará un método llamado "nombredelobjeto_tipoevento" y estará vacio. En la siguiente imagen vemos el código que será ejecutado al pulsar sobre el objeto "textBoxAngle".

```
//Selecciona el textbox Angle para introducir los datos.
private void textBoxAngle_GotFocus(object sender, EventArgs e) {
    //Limpiamos el textBox y ponemos el booleano en el array a true
    putTextBoxAngle("");
     for (int i = 0; i != 9; i++) {
        bTextBoxRow[i] = false;
    }
    //Para introducir en ese textbox los datos y cambiar el color
    bTextBoxRow[0] = true;
    cambioColor();
}
```

Como se puede observar en la siguiente imagen, para el objeto “linklabel” y su evento “Click” reutilizaremos el evento programado para el objeto “textBoxAngle”.

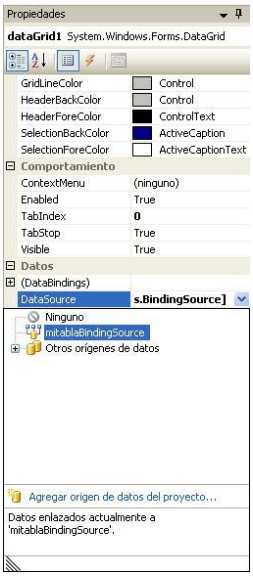


El objeto “TabControl”, además de tener las propiedades y eventos que todos los objetos tienen, hay que añadir las pestañas que se desean. Para realizar esta acción, nos dirigimos a “propiedades” y en la variable “TabPages” pulsamos sobre “…”. Aparecerá una nueva ventana en la cual podemos agregar nuevas pestañas, modificar el nombre, posición, etc.

Como se puede observar en la siguiente imagen, tenemos 3 pestañas. La primera pestaña que esta seleccionada podemos ver que es la pestaña “Table”.

Para confirurar el "DataGrid" que se encuentra en la pestaña "Table", primero hay que incluir el fichero necesario para que esta se pueda ejecutar. Por ello hay que añadir la referencia "System.Windows.Forms.DataGrid". Para indicar que campos queremos que se muestren en la tabla, hay que ir a "propiedades" y buscar la opción "DataSource".

Como se puede observar en la siguiente imagen, indicamos que los datos son "mitablaBlindignSource".



4.3. Creación de la conexión

La conexión entre el "PIC DEM FS USB" y el programa original se hace mediante USB. Por ello, se estuvo probando primero a realizar una conexión USB con las librerías disponibles. Primero se compilo el código del programa original para detectar un USB, pero había métodos no disponibles para la versión compacta.

Mas tarde, al conectar directamente la placa "PIC DEM FS USB" al dispositivo "Friendly ARM" mediante USB, Windows detectaba que algo se había conectado pero pedía los drivers para ese dispositivo. Se intento crear un driver específico para el pic, pero la información que hay sobre creación de drivers para dispositivos portátiles no es muy amplia.

Por último, al ver que ambos dispositivos tenían conectores RS232, se decidió crear una conexión serie, que fue completamente satisfactoria. Se hizo varias pruebas, con diferentes velocidades de transmisión que fueron desde los 9600 hasta los 115200 baudios, con el envió de un byte hasta 65.

Con las diferentes pruebas se llego a la conclusión de que la velocidad máxima a la que se podía enviar una serie de 65 bytes sin ninguna perdida en la transmisión era de 57600 baudios.

Después de las pruebas de velocidad, se miro el tipo de recepción y envío de los datos respecto al "PIC DEM FS USB". La recepción de datos se debía de hacer por interrupciones, ya que si llega un byte y no se ha leído el anterior, se produce un error de sobre escritura y hace falta reiniciar el "PIC DEM FS USB".

Respecto al tema del envío de datos, primero se probó por interrupciones, pero lo único que se conseguía era ralentizar las demás interrupciones. Finalmente, el envio se realizó durante la ejecución del programa.

Primero, se deben introducir los datos a enviar en un array de bytes, meter el primer dato en el registro para enviar, comprobar si el registro esta vacio y cuando este vacío el registro, introducir un nuevo byte.

También se probro la función "putsUSART()" para enviar un array de bytes. Se tiene que llamar a la función con la variable que contiene el array y el programa realizará el envio. Hubo un problema, ya que esta función piensa que la trama es un string, por lo que al intentar enviar el byte 0x00, piensa que es el carácter final de cadena y no envía mas datos.

Como el envío y recepción por RS232 no es muy fiable y los datos pueden corromperse por el ruido que hay en el entorno, se implemento un método para calcular una suma de todos los datos que enviamos. Este dato se enviara el último byte antes del fin de cadena. Si esta suma no es igual, la trama recibida será desechada.

Además de este método, ambos dispositivos desecharan todos los bytes recibidos a través del puerto si primero no recibe un byte en concreto. Con el primer byte se sabe cuando comienza la trama de datos a recibir y así comenzamos la trama en la posición 1, que será un número necesario para el comienzo de una función en concreto. Para terminar con la trama de bytes se recibe otro byte en concreto, aunque se ha pensado en ampliar este byte a varios, ya que puede darse el caso que este byte se reciba antes por los cálculos realizados y la trama no llegue correctamente.

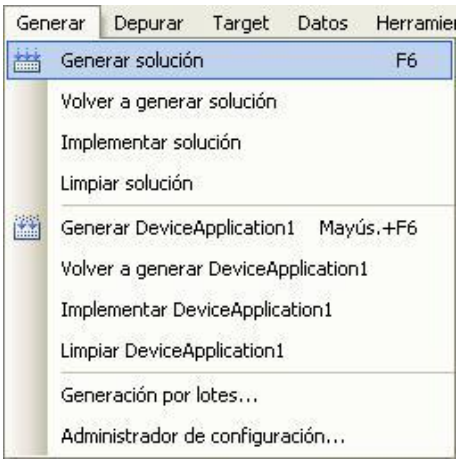
Finalmente, al recibir la trama y realizar la comprobación correctamente, se decidió que debía enviar otra trama para indicarle al "Friendly ARM" que la ha recibido correctamente y se ha ejecutado satisfactoriamente. Si esta confirmación no llega, se volverá a reenviar.

5. COMPILAR Y EJECUTAR

Una vez explicado los pasos que seguimos para crear el programa, veremos como compilar el programa en un archivo ejecutable y comprensible para el friendly.

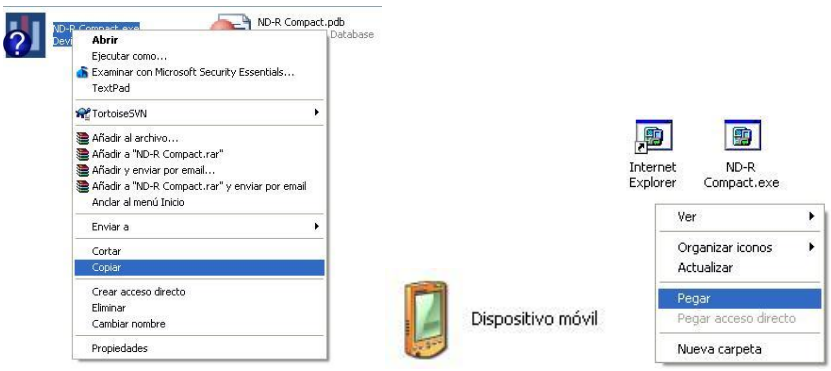
Para poder leer el dispositivo desde el ordenador de sobremesa, hace falta tener instalado el Active Sync de Microsoft que se peude descargar desde la dirección 8 del anexo 1.

Para ello, deberemos ir al menú "Generar\Generar solución". Pulsado esta opción primero comprobará errores de escritura en el código, y si no hay ningúno, comenzara con la compilación del programa.



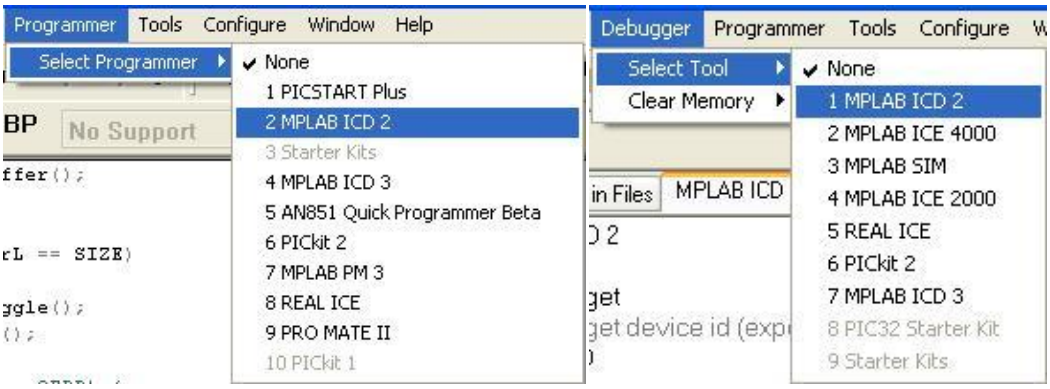
Una vez terminada, el archivo ejecutable se encuentra en la carpeta predeterminada de los proyectos en "…\nombre_proyecto\bin\debug\". Este programa solo se puede ejecutar en el dispositivo portátil, y para ello deberemos copiarlo.

Primero, buscaremos la anterior carpeta y sobre el archivo ".exe" pulsaremos con el botón secundario del ratón y seleccionaremos copiar. Segundo, en "mi pc" seleccionaremos "dispositivo móvil" y en la carpeta que queramos, en nuestro caso "Windows\Desktop", pulsaremos con el botón secundario y seleccionaremos pegar. Estos pasos se pueden observar con las siguientes imágenes.

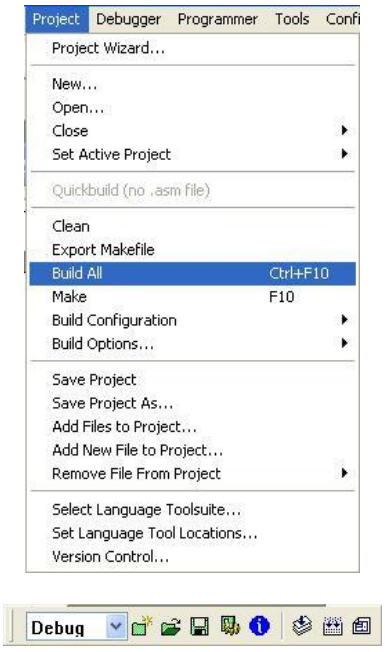


Una vez en el dispositivo portátil, el programa aparecerá en el escritorio y puslando sobre él se ejecutará. Si queremos que se ejecute al iniciar el sistema operativo copiaremos el archivo en "\Windows\StartUp".

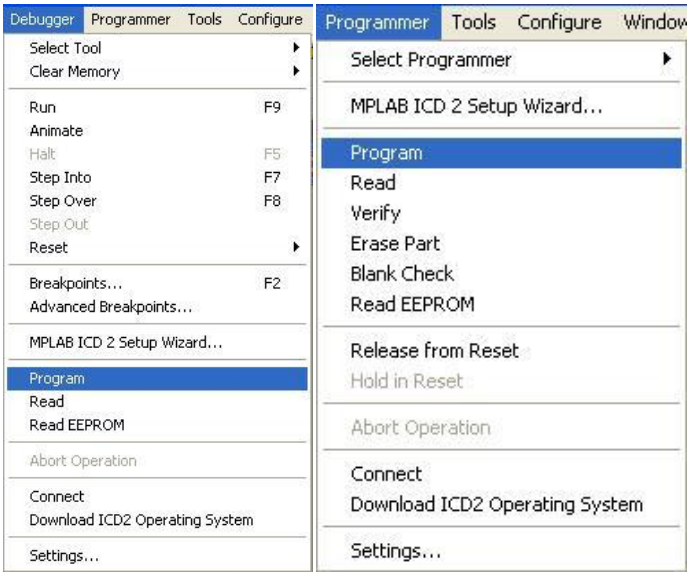
Para programar el Pic hace falta el "MPLAB". Una vez iniciado, seleccionaremos si queremos programar el dispositivo con la opción "programmer", o programar y observar las variables o estados del pic con la opción "debugger". Como se puede observar en las siguientes imágenes, seleccionamos nuestro grabador "MPLAB ICD 2".



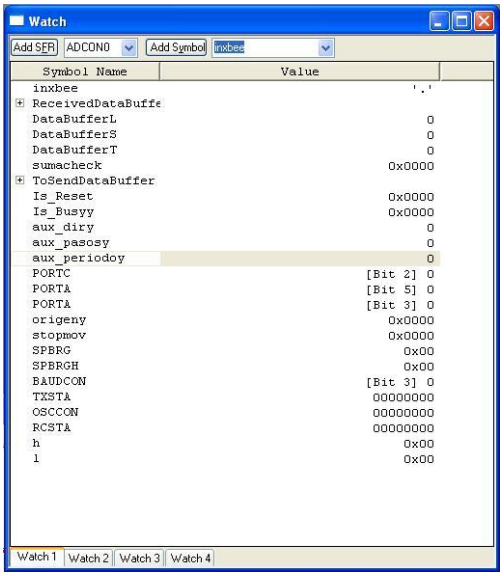
Una vez seleccionado, para compilar el programa dispondremos de botones en la barra de herramientas o en el menú "Project" pulsando la opción de "make" o "build all", o con el botón tercero o segundo comenzando por la derecha respectivamente.



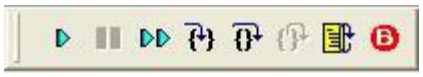
Una vez compilado el archivo, si no da ningún error en esta fase, podremos grabar el dispositivo mediante, dependiendo de la opción indicada antes, en el menú “debugger” o “programmer” seleccionado la opción “program” o pulsando el primer botón de la izquierda de siguiente barra de herramientas.



Si hemos seleccionado el modo “debugger” tendremos la posibilidad, mediante el menú “View\watch” de obtener una nueva ventana en la que se puede indicar que variables queremos observar mientras el dispositivo se ejecuta.



Para actualizar los valores de la variables tendremos que parar la ejecución del programa. Luego se puede volver a arrancar desde la última línea de código ejecutada. Estas acciones se puede realizar mediante los botones de la siguiente barra de herramientas.



Además, se puede parar la ejecución del programa cuando llege a una línea de código en concreto mediante un “break point”. Para ello solo hay que ir a la línea deseada y pulsar dos veces en ella y se indicará en la misma línea con una “b” en un circulo rojo como se puede observar en la siguiente imagen.

6. MANUAL DEL USUARIO

La aplicación creada funciona con el mismo tipo de datos que el programa original. Puede leer y escribir en ficheros XML para cargar datos en la tabla y así poder realizar las operaciones o en cambio, salvarlas para realizarlas en otro momento.

La tabla contiene 11 datos que son:

- Id: campo auto numérico. No hace falta modificarlo.
- Name: Campo indicativo para el programa original. Este campo no se puede modificar ya que no se dispone de un teclado físico o virtual para introducir caracteres.
- Angle: Indica cuantos grados se tiene que mover desde la posición inicial.
- Rotation Speed: Indica la velocidad a la que se realiza el giro.
- Dips: Indica cuantas repeticiones se va a realizar la fila.
- H Up: Indica a qué altura inicia el proceso
- H Down: Indica a qué altura termina el proceso.
- Immersion speed: Indica a qué velocidad baja el sistema robótico.
- Withdrawal speed: Indica a qué velocidad sube el sistema robótico.
- T Up: Tipo Int32. Indica cuanto tiempo estará en la posición superior.
- T Down: Tipo Int32. Indica cuanto tiempo estará en la posición inferior.

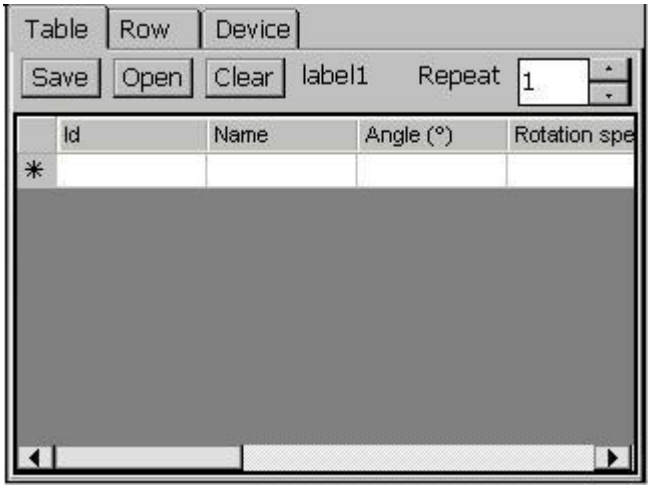
El único dato que no se introduce en la tabla es el número de repeticiones.

Explicado los datos necesarios, explicaremos la interfaz gráfica. Como comentamos en el apartado anterior, se decidió hacer la interfaz gráfica mediante pestañas. Cada una de ellas tendrá una parte del programa utilizado en el ordenador de sobremesa.

La primera pestaña se llama “Table”. Como se puede observar en la siguiente imagen, en la primera pestaña se dispone de:

- Una tabla en la cual se mostraran todos los datos. Esta vista puede ser útil para ver cómo cambia un dato respecto a las órdenes dadas.
    - Al pulsar sobre una fila, esta se mostrará en la pestaña “Row”.
- Botón “Open”. Al pulsar sobre este botón se abrirá una ventana en la cual nos dará la opción de elegir un archivo con extensión XML para abrir y cargar.
    - Si antes de abrir, el programa se ha detectado que el fichero anterior no está grabado, este nos preguntara si se desea grabar el archivo anterior, abrir uno nuevo o cancelar la operación en curso.
    - Si se selecciona un archivo que no tiene una estructura interna deseada, no cargara el archivo.
    - Si en alguna fila, los datos de cada columna no están entre los límites, mostrara un error y nos dirá cuantas filas no ha introducido en la tabla, mientras que introducirá bien los datos no erróneos.

- Botón “Save”. Al pulsar sobre él se abrirá una nueva ventana en la cual nos dejara seleccionar un archivo existente para sobrescribirlo. En cambio, si pulsamos sobre la línea de nombre del archivo, se mostrara un teclado virtual en el cual podremos introducir un nombre.
- Botón “Clear”. Al pulsar sobre él, se vaciara todos los datos de la tabla.
- Etiqueta “Label1”. Muestra el nombre del archivo cargado. Siempre se mostrara al abrir o guardar un archivo como nombre predeterminado.
- Botón numérico “Repeat”. Indica el número de repeticiones que se va a realizar los datos introducidos.

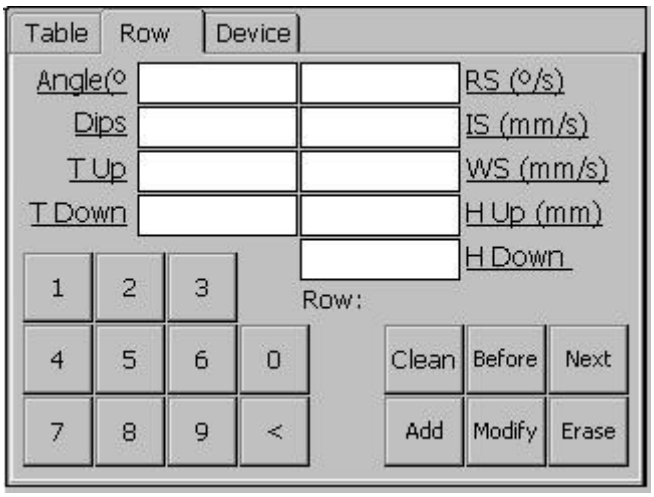


La segunda pestaña se llama Row. Como se puede observar en la siguiente imagen se dispone de:

- 10 botones comprendidos entre el 0 y el 9. Estos se irán introduciendo siempre en la última posición. Por ejemplo, si pulsamos primero el 1 y luego el 3, el número introducido será el 13.
- El botón “<” eliminara el último número introducido.
- 9 recuadros de texto para introducir en cada uno de ellos los datos pertinentes
    - Al pulsar sobre estos recuadros, o sobre la etiqueta que está a su lado, el programa pondrá el cursor sobre el recuadro y podremos introducir los números.
    - Si hay algún número introducido en ese recuadro, este se eliminará.
- Todos los recuadros, salvo uno estarán en rojo. El color blanco indica que se introducirán los números en ese recuadro.
- Al pulsar el botón “Add”, el programa añadirá los datos a la tabla.
    - Si hay algún dato incorrecto, se mostrara una ventana recordando los límites de aquellos datos en los que se ha sobrepasado.
- Al pulsar el botón “Modify”, el programa cambiara la línea seleccionada por los datos establecidos

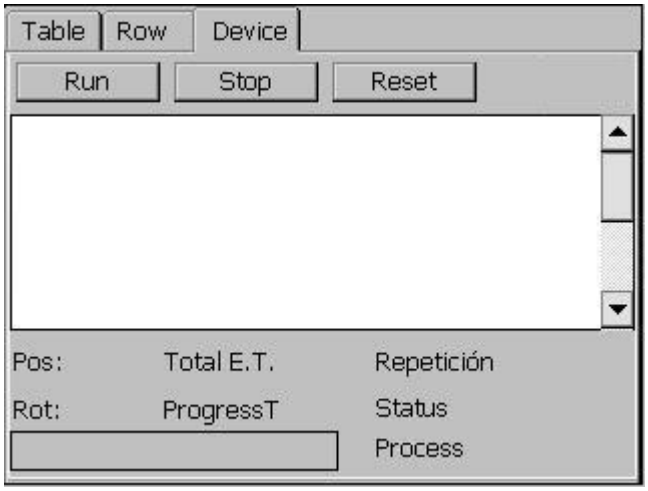
- Si no se ha modificado ningún dato, el programa nos mostrara una ventana diciendo que no se han realizado cambios.
- Si al introducir los nuevos datos, si hay algún dato incorrecto, se mostrara una ventana recordando los límites de aquellos datos en los que se ha sobrepasado.
- Si se han introducido todos los datos correctamente nos pedirá una confirmación para realizar el cambio.

- Al pulsar el botón “Erase”, el programa eliminara la fila seleccionada mostrada.
  - El programa nos pedirá una confirmación para eliminar la fila.
- Al pulsar el botón “Clean”, el programa limpiara los datos introducidos en todas los recuadros y devolverá el control al recuadro “Angle”.
- Al pulsar el botón “Before”, el programa mostrara la fila anterior en la tabla.
  - Si se está mostrando la fila número 0, se mostrara la última fila introducida.
- Al pulsar el botón “Next”, el programa mostrara la fila siguiente en la tabla.
  - Si se está mostrando la última fila, se mostrara la fila número 0.



La tercera y última pestaña se llama “Device” y como se puede observar se dispone de:

- El botón “Run” inicia el proceso de comunicación y envía los datos introducidos en las anteriores pestañas.
    - Una vez pulsado, este pasara a “Pause”. Al ser pulsado parará el proceso de envio y se quedara inactivo y cambiara la etiqueta a “Restart”.
    - Si se pulsa cuando aparece la palabra “Restart”, el programa recomenzara a enviar los datos.
- El botón “Stop” para el proceso de envió de datos.
- El botón “Reset” mandara al brazo robótico al punto de reposo.
- En el recuadro de texto se podrá observar indicaciones que el programa va mostrando.
- La etiqueta “Pos” muestra la altura en mm a la que se encuentra el sistema robótico.
- La etiqueta “Rot” muestra la rotación en grados a la que se encuentra el sistema robótico.
- La etiqueta “Repetición” muestra en la repetición que esta.
- La etiqueta “Status” muestra el estado del proceso.
- La etiqueta “Process” muestra el nombre del proceso.
- La etiqueta “Total E.T.” muestra el tiempo total estimado.
- La etiqueta “ProgressT” muestra el progreso actual del proceso.
- La barra de progreso muestra una estimación de lo que queda para terminar el estado actual.



Para el correcto funcionamiento entre el sistema “Friendly ARM” y el “Pic Dem Fs USB” la conexión serie debe ser por cable. Por ello, se ha configurado ambos dispositivos para trabajar sin tener que configurar nada por parte del usuario. Solo hay que tener en cuenta que el cable de conexión entre ambos dispositivos tiene que ser cruzado.

7. MANUAL DEL PROGRAMADOR

7.1 Configuración del Pic

Antes de empezar, se puede descargar el datasheet del PIC 18F4550 de la web que se encuentra en el anexo 1 dirección 6. Aunque se puede ver la configuración de los registros utilizados en el anexo 2.

Respecto a la configuración del Pic, debemos cambiar pequeños segmentos de código. En este caso hay que añadir un método para recibir y otro para enviar.

Para configurar la conexión serie en el pic hay que hacer lo siguiente:

- Configurar el pin RC7 como entrada
- Configurar el pin RC6 como salida
- Configurar el registro TXSTA con el valor 0x24 con lo que configuramos:
  - Habilitar el envió de datos
  - Habilitar transmisiones rápidas
  - Modo asíncrono de la transmisión
  - Transmisión de 8 bytes
- Configurar el registro RCSTA con el valor 0x90 con lo que configuramos:
  - Habilitar la recepción de datos
  - Habilitamos los pines RC7 y RC6 para la recepción y envío
- Configurar el registro BAUDCON con el valor 0x08 con lo que configuramos:
  - Habilitamos dos registros para configurar la velocidad
- Configurar el registro PIE1 con el valor 0x20 con lo que configuramos:
  - Habilitamos la interrupción por recepción
- Configurar el registro SPBRGH y SPBRG con los siguientes valores:
  - 0x04 y 0x E2 para una comunicación a 9600 baudios
  - 0x02 y 0x71 para una comunicación a 19200 baudios
  - 0x01 y 0x38 para una comunicación a 38400 baudios
  - 0x00 y 0xD6 para una comunicación a 56000 baudios
  - 0x00 y 0xD0 para una comunicación a 57600 baudios
  - 0x00 y 0x68 para una comunicación a 115200 baudios

Una vez configurado el pic para la transmisión y recepción, debemos crear los métodos para poder recibir y enviar los datos.

El envió se realiza durante la ejecución del programa:

```
void enviar(unsigned char out[SIZE], int j){
    i = 0;
    while (i < j) {
        if (PIR1bits.TXIF) {
            TXREG = out[i];
            i++;
        }
    }
}
```

Al método enviar le pasamos un array de bytes, o en este caso al no haber ese tipo de dato, utilizamos el unsigned char y un numero. Este número debe ser 1 mayor al que queremos enviar. Por ejemplo, si el array de bytes contiene 9 elementos, hay que decirle que envié 10.

En este caso, primero se comprueba si hemos enviado todos los elementos. Luego se chequea el bit TXIF del registro PIR1. Este bit indica si el registro TXREG esta vacio. En el caso de que este vacío, un nuevo byte se copia al registro y se envía. El registro TXREG es el encargado de pasar al puerto serie los 8 bits que contiene.

La recepción se realiza mediante interrupciones y este es su método:

```
if (PIR1bits.RCIF)
{
    PIR1bits.RCIF=0;
    inxbee = RCREG;
    if ((DataBufferL != -1) && (DataBufferL != SIZE))
    {
        ReceivedDataBuffer[DataBufferL] = inxbee;
        DataBufferL = DataBufferL + 1;
    }
    if ((inxbee==0x2D) && (DataBufferL == -1))
    {
        DataBufferL = 0;
    }
    else if ((inxbee==0x23) && (DataBufferL != -1))
    {
        DataBufferT = DataBufferL;
        DataBufferL = SIZE;
    }
}
```

Lo primero que hacemos al recibir la interrupción, es desactivarla. Sacamos el dato del registro y comprobamos si el dato recibido es el inicio de cadena o el de final. Si se ha recibido el carácter de inicio, iniciamos la variable “DataBufferL” a 0 para indicar que todos los datos después del recibido hay que guardarlo en el array de bytes. Una vez recibido el carácter de fin, indicamos que la variable “DataBufferT” es igual a la variable “DataBufferL”, para que así, si deseamos realizar alguna función con el array, solo realice la función entre los valores recibidos. Ponemos “DataBufferL” al tamaño máximo para indicar que se ha recibido los datos y se puede realizar el proceso que se desea.

La recepción, hace falta realizarla mediante interrupciones ya que se debe actuar rápidamente para que no llegue otro dato sin sacar el anterior. Si esto ocurriera, se activa el bit 1 del registro RCSTA, que indica que se ha producido un fallo en la recepción.

7.2 Aplicación

Aquí se explicara los diferentes archivos que compone el programa original. Para empezar se creó una clase para la comunicación serie, otra clase para lanzar excepciones personalizas y otra clase que contiene la interfaz gráfica.

7.2.1. Serie.cs

La comunicación serie se puede observar en el archivo "Serie.cs". Esta clase contiene dos variables globales. La primera es la clase "Serial Port" con la que accederemos a la configuración del puerto serie. La segunda es un booleano.

Para configurar la velocidad, tipo de conexión, número de bytes a enviar o recibir, tenemos en el constructor de esta clase. Por ahora solo modificaremos la velocidad y las demás variables, al estar comentadas, cogerán la configuración por defecto.

```
public Serie() {

    try {
        sp = new SerialPort();

        sp.BaudRate = 57600;

        //sp.BaudRate = 9600;
        //sp.Parity = Parity.None;
        //sp.StopBits = StopBits.One;
        //sp.DataBits = 8;
        //ps.Handshake = Handshake.None;

        sp.Open();
    }

    catch (Exception) {
        MessageBox.Show("Error Serie.cs Serie()");
    }
}
```

Una vez configurada la conexión, hay que realizar un semáforo para que los métodos encargados de enviar datos no utilicen el puerto serie a la vez. Para ello, con la variable booleana podemos simular un semáforo. Si el booleano es verdadero, quiere decir que hay un método que está utilizando el puerto serie por lo que deberá esperar el nuevo método hasta que se haya enviado todos los datos.

```
//Liberamos el control
public void liberar() {
    busy = false;
}

//Ocupamos el control
public void ocupado() {
    busy = true;
}

//Devolvemos el estado del control
public bool IsOcupado() {
    return busy;
}
```

Para enviar por el puerto serie se crearon varios métodos con diferentes tipos de variables, pero el que utilizaremos es el método que envía un array de bytes. Para ello, hay que pasar dos variables al método. La primera es el array de bytes a enviar y la segunda es el número de bytes a enviar. Una vez dentro del método, este llama a la variable "Serial Port" y envía desde el byte 0 hasta el valor de la segunda variable del array de bytes.

```
public bool enviarDato(Byte[] OUTBuffer,int size) {
    try {
        sp.Write(OUTBuffer, 0, size);
        return true;
    }
    catch (Exception) {
        MessageBox.Show("Error Serie.cs enviarDato(byte[], int)");
        return false;
    }
}
```

Para recibir por el puerto serie es bastante parecida al método de recepción del pic. Comprobamos si el primer byte es el 0x2D. Si este es el caso leerá todos los bytes recibidos hasta que llegue el byte 0x23. Si se produce alguna excepción en este método, devolverá un array nulo.

```
public Byte[] recibirDato() {
    Byte[] buffer = new Byte[65];
    Byte leido;
    int i = 0;
    try {
        if (sp.BytesToRead != 0) {
            leido = Convert.ToByte(sp.ReadByte());
            if (leido == 0x2d) {
                while (leido != 0x23) {
                    buffer[i] = leido;
                    i++;
                    leido = Convert.ToByte(sp.ReadByte());
                }
                buffer[i] = leido; //ultimo caracter #
                return buffer;
            }
            else {
                sp.DiscardInBuffer();
                return null; //Si el primer caracter no es correcto vaciamos el buffer
            }
        }
        else return null; //Si no hay nada que leer
    }
    catch (Exception) {
        MessageBox.Show("Error Serie.cs recibirDato()");
        return null;
    }
}
```

7.2.2. MenuPrincipalMini.cs

Esta clase dispone de una gran cantidad de líneas que hay que comentar y especificar. Por ello vamos a comenzar a explicar el método constructor y las variables que dispone esta clase.

Comenzamos con las variables utilizadas:

- bTextBoxRow [9]: array de 9 booleanos con los que controla la introducción de datos en los 9 textBox de la pestaña Row. El único valor verdadero que está en este array, será el encargado de introducir los datos en el textbox.
- Send[65]: array de 65 bytes que encargada de guardar la última función enviada.
- ISend: variable del tipo int que indica cuantos bytes han sido utilizados en la última función enviada.
- Conexión: variable que contiene una instancia de la clase Serie.
- Guardado: variable del tipo booleano que representa si el archivo ha sido modificado
- NumFilaTotal: variable del tipo int que indica el número de filas totales que se han introducido en la tabla
- NumFila: variable del tipo int que indica que fila se está mostrando en la pestaña Row.
- Envio: variable utilizada del tipo int para indicar que función es la siguiente ha enviar.

Las demás declaradas son copiadas del programa original.

El método constructor, aparte de inicializar los componentes como botones, textbox y demás herramientas gráficas añadidas a la interfaz, realiza:

- La instancia de la clase Conexión
- Indica a la etiqueta “Archivo” el nombre “new file”.
- Inicializa la variable guardado a verdadero.

Una vez terminado con las variables y el método constructor, comenzamos a explicar los diferentes métodos que se utilizan en la pestaña “Table”

- **public void putArchivo(string):** Cambia el valor de "Text" de la etiqueta "archivo" por el valor pasado por la variable.
- **public string getArchivo():** Obtiene el valor de la variable Text de la etiqueta "archivo". La etiqueta "archivo" sirve para ver el nombre del archivo actual cargado en el programa. Si lleva un * quiere decir que se ha modificado.
- **private void dataGrid1_Click():** Evento generado al pulsar en la tabla "DataGrid". Se selecciona la fila completa y se muestra los datos de la misma en la pestaña Row.
- **private void buttonClear_Click():** Evento generado al pulsar el boton "buttonClear".
- Elimina todos los datos almacenados en la tabla. Si detecteca que no esta guardado, nos pedirá confirmación para continuar.
- **private void buttonOpen_Click()**: Evento generado al pulsar el boton "buttonOpen". Llama a la funcion Abrir().
- **private void Abrir():** Abrirá una nueva ventana para seleccionar el archivo. En esta función se comprueba si es un archivo XML, y además, chequea los datos antes de pasarlos a la tabla. Si hay alguna fila errónea, muestra el número de la fila errónea y borra la misma.
- **private void abrirxml(String)**: Abre y lee el archivo pasado por la variable del tipo texto.

- **private void buttonSave_Click():** Evento generado al pulsar el boton "buttonSave". Llama a la función Guardar().
- **private void Guardar():** Abrirá una nueva ventana para introducir donde quieres guardar el archivo.
- **private void guardarxml(string):** Genera el archivo XML con el nombre pasado por la variable del tipo texto.

Una vez terminado con los métodos utilizados en la pestaña “Table”, comenzamos a explicar los diferentes métodos que se utilizan en la pestaña “Row”.

- **private void mostrar(int)**: Muestra la fila int en los textBox de la pestaña Row y pone el cursor en el textBoxAngle
- **private void putLabelRow(int)**
- **private void putLabelRow(string)**: Cambia el valor de "Text" de la etiqueta "labelRow" por el valor pasado por la variable. Se muestra en la pestaña Row para saber que fila se esta mostrando en cada momento. X si no se ha añadido la fila o no se ve ninguna.
- **private void clear():** Elimina todos los valores de la variable "Text" de los textBox que aparecen en la pestaña Row. Además, modifica las variables para introducir directamente los datos en el "textBoxAngle".
- **private void cambioColor()**: Chequea las variables para poner en color blanco el textBox seleccionado para introducir los datos y los demas en color rojo.
- **private string limites():** Comprueba si los valores introducidos en los textBox de la pestaña "Row" estan dentro de programados. Si están dentro de los limites devolverá: "". Si hay algún dato que sale de los límites devolverá una indicación de los valores fuera de los limites. Si se produce algún error devolverá: "Error"
- **private void buttonX_Click():** Añade al final del textBox que esté en blanco el numero X. Donde X esta entre 1 y 10, siendo 10 el número 0.
- **private void button12_Click():** Elimina el último número introducido en el textBox que está en blanco.
- **private void buttonClean_Click():** Evento generado al pulsar el botón "buttonClean". Llama a la función clear().
- **private void buttonAdd_Click():** Evento generado al pulsar el botón "buttonAdd". Si los valores introducidos en los textBox son correctos, añade esos valores a la tabla.
- **private void buttonModify_Click():** Evento generado al pulsar el botón "buttonModify". Comprueba si se han realizado cambios en los valores de los textBox de la fila número X. Si se han realizado, chequea si se están entre los límites y si está correcto, modifica la fila y limpia los textBox
- **private void buttonErase_Click():** Evento generado al pulsar el botón "buttonErase". Elimina la fila número X pidiendo confirmación.
- **private void buttonNextButton_Click():** Evento generado al pulsar el botón "buttonNextButton". Muestra la fila siguiente a la actual mostrada en la pestaña Row
- **private void buttonBeforeButton_Click():** Evento generado al pulsar el botón "buttonBeforeButton". Muestra la fila anterior a la actual mostrada en la pestaña Row
- **private void textBoxX_GotFocus():** Eventos generados al pulsar sobre la etiqueta o textBox. Limpia la variable "Text" del textBox de la pestaña Row y pone en el cursor para introducir datos en el textBox seleccionado.
- **public void putTextBoxX(string):** Modifica la variable "Text" del textBox cambiándola por las variables pasadas.
- **public void putTextBoxX(int):** Modifica la variable "Text" del textBox cambiándola por las variable de la fila deseada según el número introducido.

- **public string getTextBoxX():** Devuelve el un string con el valor guardado en la variable "Text" de los textBox de la pestaña Row.
- **private Variable getX():** Devuelve en el formato requerido para introducir los datos en la tabla. Si hay algún error devuelve -1 (Fuera de rango).

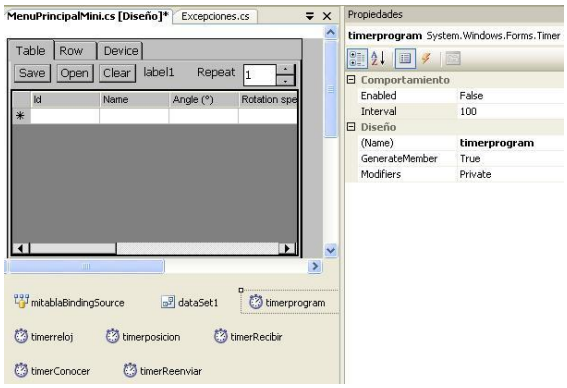
Una vez terminado con los métodos utilizados en la pestaña “Row”, comenzamos a explicar los diferentes métodos que se utilizan en la pestaña “Device”.

- **public void putLabelX(string):** Cambia el valor de "Text" de la etiqueta "labelX" por el valor pasado por la variable.
- **private void buttonRun_Click():** Evento generado al pulsar el boton "buttonRun". Llama a la funcion AccionButtonRun();
- **private void buttonStop_Click():** Evento generado al pulsar el boton "buttonRun". Llama a la funcion AccionButtonStop();
- **private void buttonReset_Click():** Evento generado al pulsar el boton "buttonRun". Llama a la funcion AccionButtonReset();
- **private void AccionButtonRun():** Metodo copiado del programa original
- **private void AccionButtonStop():** Metodo copiado del programa original
- **private void AccionButtonReset():** Metodo copiado del programa original
- **public string getTextBoxDevice():** Devuelve el valor de la variable "Text" del textBox que está en la pestaña device
- **public void addTextBoxDevice(string):** Añade a la variable "Text" del textBox de la pestaña Device el valor pasado
- **public void limpiarTextBoxDevice():** Pone el valor "" en la variable "Text" del textBox de la pestaña Device.

Una vez terminado con los métodos utilizados en la pestaña “Device”, comenzamos a explicar los diferentes métodos que se utilizan en el “DataSet”.

- **public int getNumberOfRow():** Nos devuelve el número de las filas totales que hay.
- **public DataRow getDataRow(int):** Devuelve los valores de la fila pasada por la variable
- **private void putDataX(int,Variable):** Introduce en la fila int de la tabla el valor de "Variable".
- **private Variable getDataX(int):** Devuelve los valores de la fila int de la tabla

Explicaremos ahora los métodos utilizados con los temporizadores. Para cambiar el tiempo de ejecución de cada temporizador, hay que ir a la interfaz gráfica, pulsar en el temporizador deseado y en la pestaña de propiedades modificar el campo “Interval”.



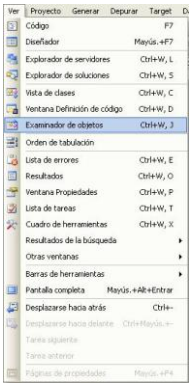
- **private void timerConocer_Tick():** Evento generado por un reloj. Ejecuta el método AccionConocer()
- **private void timerRecibir_Tick():** Evento generado por un reloj. Ejecuta el método AccionRecibir()
- **private void AccionConocer():** Envía a través del puerto serie dos funciones, la 0x37 y 0x81
- **private void AccionRecibir():** Si hay un dato en el buffer de recepción RS232, traduce lo recibido para que en el caso 0x37 conozca la posición del robot y en el caso 0x81 conozca el estado del pic
- **private void timerreloj_Tick():** Activa y desactiva temporizadores y llama al método AcciónReloj
- **private void timerposición_Tick():** Activa y desactiva temporizadores y llama al método accionPosición
- **private void AcciónPosición():** Método copiado del programa original. Se ha modificado las variables donde se escriben las indicaciones
- **private void AccionProgram():** Método copiado del programa original. Se ha modificado las variables donde se escriben las indicaciones
- **private void AccionReloj():** Método copiado del programa original. Se ha modificado las variables donde se escriben las indicaciones
- **private void reenviar(Byte[],int):** Copia la ultima función enviada.
- **private void enviar(Byte[],b):** Envía el dato deseado a través del puerto serie. Este controla si el puerto serie esta en uso o libre.
- **private bool recibir():** Solo recibe la cadena "-Ok#" para desactivar el control de reenviar

Terminado con los temporizadores, explicaremos los tres métodos que se utilizan para dar órdenes al pic.

- **public void Sub(long,int,uint):** Método copiado del programa original. Modificado para enviar un Checksum y por puerto RS232
- **public void Girar(long,int,uint):** Metodo copiado del programa original. Modificado para enviar un Checksum y por puerto RS232
- **public void Reset():** Metodo copiado del programa original. Modificado para enviar un Checksum y por puerto RS232

Hay otros métodos o clases que se utilizan para la realización del programa que se encuentran en las librerías del programa, como por ejemplo la clase "MessageBox". Desde el propio programa o desde la web se puede buscar su funcionamiento.

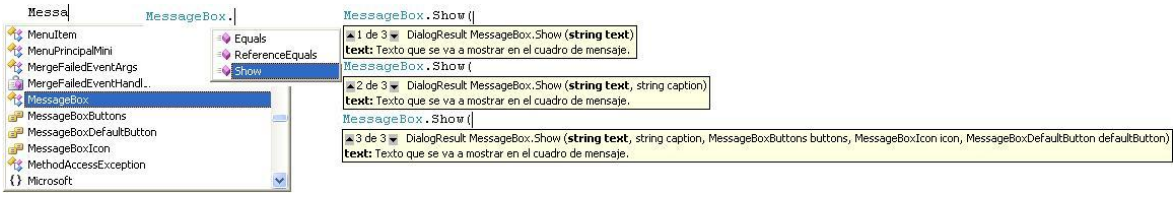
Desde el programa se puede observar los métodos que se pueden utilizar desde el "Examinador de Objetos". Para acceder a este apartado, debemos pulsar en el menú "Ver/Examinador de Objetos".



En la nueva ventana que aparecerá debemos introducir en "Examinar" la opción "Mi solución", así solo buscara en las librerías que tenemos añadidas a nuestro proyecto. Debajo de este hay un recuadro donde introduciremos el método a buscar. Una vez buscado nos indicara todos aquellos métodos, con sus sobrecargas que se pueden utilizar y el tipo de datos que debemos introducir. También se indica en que librería se encuentra, por si se quiere añadir algunas nuevas.

En el editor de texto también se puede obtener una ayuda a la hora de escribir los métodos utilizados por las librerías. Cuando se empieza a escribir nos aparece una pequeña ventana con todos los datos, variables y métodos que podemos utilizar en nuestro proyecto. Al ir escribiendo letras se hace una búsqueda.

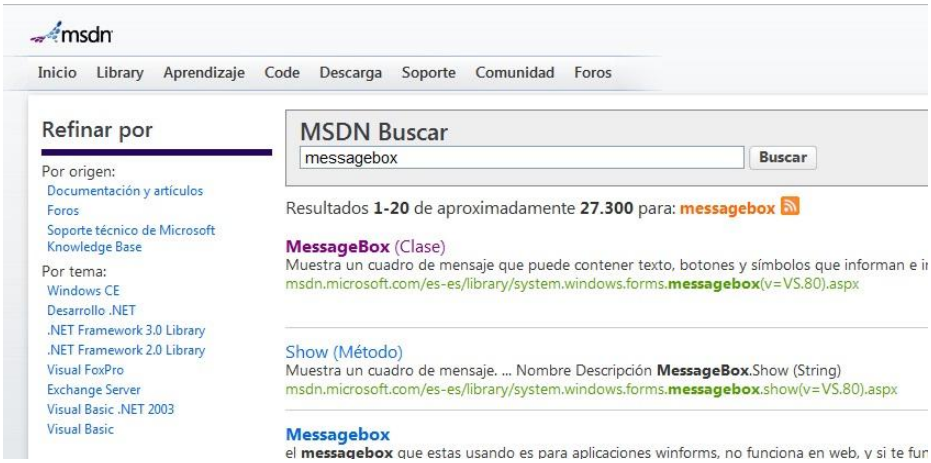
Como se puede observar en la siguiente imagen, al escribir "Messa" nos indica la posibilidad de "MessageBox" y al pulsar la tecla "Enter" y escribir el "." aparecen los métodos disponibles. Pulsando otra vez "Enter" y escribiendo en este caso "(" nos aparece una ayuda con los datos que se deben pasar al método. Al haber varias sobrecargas podemos ver todos pulsando las teclas de dirección arriba y abajo. Además, se marca en negrita el dato que debes introducir actualmente.



Por último, también se dispone las librerías a través de internet. La web esta en el anexo 1 dirección 7.

Desde la página de internet, iniciaremos la búsqueda introduciendo el nombre de la clase que queremos buscar.

Esta realizara una búsqueda, sobre las bases que tiene. Del resultado se tiene que pinchar sobre, en este caso, “MessageBox (Clase)”.



Al pulsar sobre la clase entraremos en su apartado y nos mostrarán ejemplos de utilización y de código. Antes de seguir avanzando debemos elegir la versión del .NET Framework correspondiente.

Para ver los métodos, nos dirigiremos al lado izquierdo de la página, donde pulsaremos sobre “MessageBox (Métodos)”. Para saber qué métodos están disponibles para la versión “Compact”, observamos que en la primera columna aparecen unos símbolos. El símbolo de “teléfono” indica que está disponible para la versión “Compact”.

7.3 Comunicación

En ambas aplicaciones se han creado las mismas variables con el mismo tamaño para facilitar la comunicación entre ellas.

Para empezar dispondremos de dos buffers diferentes, uno para enviar y otro para recibir la información. Ambos son de tamaño byte y de 65 de longitud. Aunque sean de tamaño 65, solo enviaremos aquellos bytes necesarios.

Empezaremos siempre con el byte 0x2D en hexadecimal, posición 0. Con esto indicaremos que comienza la transmisión en ambos sentidos. Para terminar la transmisión, posición n, siempre se hará con el byte 0x23.

El segundo byte, posición 1, será la función que identificará los datos a enviar y son las siguientes. En el Pic no se guardar el byte 0x2D, para que el primer byte guardado sea la función y así no modificar mas el programa original.

- 0x37: Devuelve al "Friendly" los siguientes datos: del byte 2 al 4 se envía la variable "distanciaX", del byte 5 al 7 se envía la variable "distanciaY".
- 0x80: Alterna el estado del led número 1 y número 2.
- 0x81: Devuelve al "Friendly" los siguientes datos: en el byte 2 devuelve el booleano posunknown, en el byte 3 devuelve la variable Is_BusyX, en el byte 4 devuelve la variable Is_BusyY, en el quinto byte devuelve la variable Is_Reset
- 0x85: realiza la operación de movimiento para el eje X con un periodo menor a 60 microsegundos.
- 0x90: realiza la operación de movimiento para el eje X con un periodo mayor que 60 microsegundos.
- 0x92: realiza la operación de movimiento para el eje Y.
- 0x96: para los procesos.
- 0x98: realiza la función de reset.

Antes de terminar la transmisión, para comprobar la integridad de los datos, desde el byte número 2 hasta n-2 se calculara la suma. A esta suma se le aplicara una máscara del tipo 0x00FF, para que el resultado solo sea de un byte, que irá en la posición n-1.

0	1	2 … n-2	n-1	n	… 65
0x2D	Función	Datos a enviar	Checksum	0x23	0x00

En ambos dispositivos, para calcular el checksum, se hace con el siguiente código. El primer código corresponde al programa del Pic y el segundo corresponde al programa del "Friendly".

```
unsigned char checksum(unsigned char c[], int a, int b){
    sumacheck = 0;
    for (i = a; i < b; i++) sumacheck += c[i];
    return (sumacheck)&0x00FF;
}
private Byte checksum(Byte[] c, int a, int b) {
    int sumacheck = 0;
    for (int j = a; j < b; j++) sumacheck += c[j];
    return Convert.ToByte((sumacheck) & 0x00FF);
}
```

Como se puede comprobar, en ambos casos, se llama a la función con 3 datos. El primer dato es el array de bytes. El segundo es la posición inicial desde la que calcula la suma. El tercer dato es la posición final hasta que queremos que sume. Para terminar se devuelve el dato de la suma aplicado a una multiplicación del tipo 0x00FF, dejando solo el número del primer byte.

ANEXO 1: DIRECCIONES WEB

Dirección 1: http://code.google.com/p/friendlyarm/wiki/EnglishBSP

Dirección 2: http://friendlyarm.googlecode.com/files/MINI2440BSP.zip

Dirección 3: http://www.friendlyarm.com

Dirección 4: http://msdn.microsoft.com/es-es/library/2weec7k5(v=vs.90).aspx

Dirección 5: http://www.microsoft.com/downloads/es-es/details.aspx?FamilyID=E3821449-3C6B-42F1-9FD9-0041345B3385

Dirección 6: http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf

Dirección 7: http://msdn.microsoft.com/es-es/library

Dirección 8: http://www.microsoft.com/downloads/es-es/details.aspx?FamilyID=9e641c34-6f7f-404d-a04b-dc09f8141141

ANEXO 2: DOCUMENTACIÓN PIC

Registro BAUDCON

REGISTER 20-3: BAUDCON: BAUD RATE CONTROL REGISTER

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **ABDOVF**: Auto-Baud Acquisition Rollover Status bit
1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6 **RCIDL**: Receive Operation Idle Status bit
1 = Receive operation is Idle
0 = Receive operation is active

bit 5 **RXDTP**: Received Data Polarity Select bit
Asynchronous mode:
1 = RX data is inverted
0 = RX data received is not inverted
Synchronous modes:
1 = CK clocks are inverted
0 = CK clocks are not inverted

bit 4 **TXCKP**: Clock and Data Polarity Select bit
Asynchronous mode:
1 = TX data is inverted
0 = TX data is not inverted
Synchronous modes:
1 = CK clocks are inverted
0 = CK clocks are not inverted

bit 3 **BRG16:** 16-Bit Baud Rate Register Enable bit
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN**: Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
0 = Baud rate measurement disabled or completed
Synchronous mode:
Unused in this mode.

Registro PIE1

REGISTER 9-6: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SPPIE:** Streaming Parallel Port Read/Write Interrupt Enable bit(1)
1 = Enables the SPP read/write interrupt
0 = Disables the SPP read/write interrupt

bit 6 **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE:** EUSART Receive Interrupt Enable bit
1 = Enables the EUSART receive interrupt
0 = Disables the EUSART receive interrupt

bit 4 **TXIE:** EUSART Transmit Interrupt Enable bit
1 = Enables the EUSART transmit interrupt
0 = Disables the EUSART transmit interrupt

bit 3 **SSPIE:** Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Note 1: This bit is reserved on 28-pin devices; always maintain this bit clear.

Registro PIR1

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
SPPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SPPIF:** Streaming Parallel Port Read/Write Interrupt Flag bit(1)
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF:** EUSART Receive Interrupt Flag bit
1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read)
0 = The EUSART receive buffer is empty

bit 4 **TXIF:** EUSART Transmit Interrupt Flag bit
1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written)
0 = The EUSART transmit buffer is full

bit 3 **SSPIF:** Master Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 2 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1: This bit is reserved on 28-pin devices; always maintain this bit clear.

Registro RCSTA

REGISTER 20-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

Registro TXSTA

REGISTER 20-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-Bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed
Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data
Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode with the exception that SREN has no effect in Synchronous Slave mode.

Recordar que la documentación completa de este pic, se encuentra disponible en internet mediante la dirección web del anexo 1 número 6.

ANEXO 3: CÓDIGO FRIENDLY ARM

Serie.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO.Ports;
using System.Threading;
using System.Windows.Forms;

namespace DeviceApplication1 {
   class Serie{
      private SerialPort sp;
      private bool busy = false;

      public Serie() {
         try {
            sp = new SerialPort();

            sp.BaudRate = 57600;

            //sp.BaudRate = 9600;
            //sp.Parity = Parity.None;
            //sp.StopBits = StopBits.One;
            //sp.DataBits = 8;
            //ps.Handshake = Handshake.None;

            sp.Open();
         }

         catch (Exception) {
            MessageBox.Show("Error Serie.cs Serie()");
         }
      }

      #region Semaforo para enviar

      //Liberamos el control
      public void liberar() {
         busy = false;
      }

      //Ocupamos el control
      public void ocupado() {
         busy = true;
      }

      //Devolvemos el estado del control
      public bool IsOcupado() {
         return busy;
      }

      #endregion


      #region Enviar,Recibir y vaciar Buffer

      public bool enviarDato(string sms) {
         try {
            sp.Write(sms);
            return true;
            //else MessageBox.Show("SMS mayor que 50");
         }
         catch (Exception) {
            MessageBox.Show("Error Serie.cs enviarDato(string)");
            return false;
         }
      }
```

```
    /*
     * Debe empezar el OUTBuffer por el byte 0x2D = '-'
     * y terminar por el byte 0x23 = '#'
     * Ambos caracteres no se cogen, por lo que en el otro programa,
     * el byte 1 de aqui es el byte 0 del PIC.
     * Size = al número de bytes a enviar mas uno.
     */

    public bool enviarDato(Byte[] OUTBuffer,int size) {
      try {
        sp.Write(OUTBuffer, 0, size);
        return true;
      }
      catch (Exception) {
        MessageBox.Show("Error Serie.cs enviarDato(byte[], int)");
        return false;
      }
    }

    public Byte[] recibirDato() {
      Byte[] buffer = new Byte[65];
      Byte leido;
      int i = 0;
      try {
        if (sp.BytesToRead != 0) {
          leido = Convert.ToByte(sp.ReadByte());
          if (leido == 0x2d) {
            while (leido != 0x23) {
              buffer[i] = leido;
              i++;
              leido = Convert.ToByte(sp.ReadByte());
            }
            buffer[i] = leido; //ultimo caracter #
            return buffer;
          }
          else {
            sp.DiscardInBuffer();
            return null; //Si el primer caracter no es correcto vaciamos el buffer
          }
        }
        else return null; //Si no hay nada que leer
      }
      catch (Exception) {
        MessageBox.Show("Error Serie.cs recibirDato()");
        return null;
      }
    }

    public void vaciarBuffer() {
      try {
        sp.DiscardInBuffer();
      }
      catch (Exception) {
        MessageBox.Show("Error Serie.cs vaciarBuffer()");
      }
    }

    #endregion

  }//Fin clase
}
```

MenuPrincipalMini.cs

```
using System;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using System.IO;
using System.Threading;


namespace DeviceApplication1 {
   public partial class MenuPrincipalMini : Form {

      delegate void SetTextCallback(string text);

      /*private string[] tipos = new string[9] {"Angle (º)", "RS (º/s)",
         "Dips", "H Up (mm)", "H Down (mm)", "IS (mm/s)",
         "WS (mm/s)", "T Up", "T Down"};*/
      private bool[] bTextBoxRow = new bool[9] {true,false,false,false,false,false,
                                    false,false,false};
      private Byte[] send = new Byte[65];
      private int numFilaTotal = 0, numFila = 0, envio = 0, isend;
      private static Serie conexion;
      private bool guardado; //True = guardado; False = no guardado

      //Variables copiadas del DIPDR
      private char[] delimiterChars = { '\\', '.' };
      private double avancepasgiro = 35.55, avancepasalt = 657.08, cvelgiro = 28125, cvelalt = 1521.87;
      private static bool Is_Reset = false, Is_Busyx = false, Is_Busyy = false;
      private int fase = 0, numrepeat = 0, inicio = 0, numprocess = 0;
      private bool boolproceso = false, iniciclo = true, tmovimiento = false, pauseprog = false;
      private int angle, posh1, posh2, speedgiro, speedsub, time1, time2, numdip, repeatotal;
      uint hpos, hpos_anterior = 0, hposy, hpos_anteriory = 0;
      uint periodox, periodoy;
      uint tiempoproctot, tiempopropar, timespera = 0;
      public static bool imanual = true, abrirproy = false, menucerrar = false, posunknown = true, asaveas = false, asave = false, irun =
true, ipause = false, istop = false;
      public static bool acreset = false, ireset = false, imov = false, imanualac = false, acsavep = false, acpasarprog = false, acpause =
false, acrun = false, acstop = false;
      bool first = true;
      public static uint pasosdev = 0, pasosdevy = 0;


      //Constructor
      public MenuPrincipalMini() {
         InitializeComponent();
         cambioColor();

         //Inicializamos variables
         putArchivo("New File");
         conexion = new Serie(); //Creamos la conexion RS232
         guardado = true;
      }

      #region Pestaña Table

      //Pone el nombre del archivo
      public void putArchivo(string sms) {
         archivo.Text = sms;
      }

      //Recupera el nombre del archivo
      public string getArchivo() {
         return archivo.Text;
      }
```

```
    //Si pulsamos en la tabla y seleccionamos una fila, muestra la fila en la pestaña row
    private void dataGrid1_Click(object sender, EventArgs e) {
      try {
        int fila = dataGrid1.CurrentRowIndex;
        if (fila != -1) {
          numFila = fila;
          dataGrid1.Select(numFila);
          mostrar(numFila);
        }
        else {
          clear();
          numFila = 0;
        }
      }
      catch (Exception) {
        MessageBox.Show("DataGrid1_Click()", "Error", System.Windows.Forms.MessageBoxButtons.OK,
        System.Windows.Forms.MessageBoxIcon.Exclamation,
        System.Windows.Forms.MessageBoxDefaultButton.Button1);
      }
    }

    //Borra todos los elementos de la tabla y de la pestaña Row y pone el cursos en Angle.
    private void buttonClear_Click(object sender, EventArgs e) {
      if (!guardado) {
        DialogResult dr = MessageBox.Show("The filas has changed, do you want save it?", "Attention",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1);
        if (dr == DialogResult.Yes) Guardar();
        else if (dr == DialogResult.No) {
          dataSet1.mitabla.Clear();
          numFila = 0;
          numFilaTotal = 0;
          clear();
          putArchivo("New File");
          guardado = true;
        }
        //else Cancel
      }
      else {
        dataSet1.mitabla.Clear();
        numFila = 0;
        numFilaTotal = 0;
        clear();
        putArchivo("New File");
        guardado = true;
      }
    }

    //Lee de la ruta especificada el archivo.
    private void buttonOpen_Click(object sender, EventArgs e) {
      Abrir();
    }

    //Guarda en la ruta especificada las operaciones que estan en la tabla
    private void buttonSave_Click(object sender, EventArgs e) {
      Guardar();
    }

    private void Abrir() {
      int fail = 0, i;
      string fichero;
      string[] split;
      try {
        //Si detecta que ha sido modificado, nos pedira si queremos guardarlo.
        if ((!guardado) && (MessageBox.Show("The file has changed, do you want save it?", "Attention", MessageBoxButtons.YesNo,
MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) == DialogResult.Yes)) Guardar();

        //Abrimos la ventana para cargar un nuevo archivo
        OpenFileDialog DialogOpen = new OpenFileDialog();
        DialogOpen.Filter = "XML file (*.xml)|*.xml";
        DialogOpen.InitialDirectory = "\\My Documents\\";
        //Si pulsamos en OK para abrir, borra los anteriores datos de la tabla y lee los nuevos
```

```
        if (DialogOpen.ShowDialog() == DialogResult.OK) {
          dataSet1.mitabla.Clear();
          fichero = DialogOpen.FileName;
          abrirxml(fichero);
          split = fichero.Split(delimiterChars);
          putArchivo(split[split.Length - 2]);

          //Comprobamos si las lineas leidas son correctas y no contienen ningun error. Si contiene se borra
          i = numFilaTotal;
          for (numFila = 0; numFila < numFilaTotal; numFila++) {
            mostrar(numFila);
            if (String.Compare(limites(), "") != 0) {
              fail++;
              dataSet1.mitabla.Rows[numFila].Delete();
              numFilaTotal--;
            }
          }
          //Muestra el mensaje por pantalla de lineas erroneas
          if (fail != 0) {
            MessageBox.Show("There are " + fail.ToString() + " rows whit erros of " + i.ToString() + ". These rows will be deleted"
              , "Attention", MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
            clear();
          }
          //Si esta todo correcto se muestra la fila número 0 en la pestaña Row
          else {
            mostrar(0);
            numFila = 0;
          }
          guardado = true;
        }
      }
      catch (Exception) {
        MessageBox.Show("File Error");
        putArchivo("New File");
        guardado = true;
      }
    }

    private void Guardar() {
      string fichero;
      string[] split;
      //Mostramos la ventana de guardado con su configuración
      SaveFileDialog DialogSave = new SaveFileDialog();
      DialogSave.Filter = "XML file (*.xml)|*.xml";
      //DialogSave.Filter = "XML file (*.xml)|*.xml|All files (*.*)|*.*";
      DialogSave.InitialDirectory = "\\My Documents\\";
      //por si hemos abierto el archivo, se pone el nombre que estaba
      fichero = getArchivo();

      //Para establecer el nombre del archivo que se abrio.
      if (!guardado) fichero = fichero.Substring(1, fichero.Length - 1);
      DialogSave.FileName = fichero;

      if (DialogSave.ShowDialog() == DialogResult.OK) {
        fichero = DialogSave.FileName; //Por si cambia el nombre o algo
        split = fichero.Split(delimiterChars);
        //Comprobamos si contiene la extension xml, y sino esta se añade
        if (System.String.Compare(split[split.Length - 1], "xml") != 0) {
          fichero = DialogSave.FileName + ".xml";
          putArchivo(split[split.Length - 1]);
        }
        else putArchivo(split[split.Length - 2]);
        guardarxml(fichero);
        guardado = true;
      }
    }

    //Metodo copiado
    private void abrirxml(string fichero) {
      //leer el xml cuando le das a abrir tabla y al iniciar el form
      int countrow = -1;
      //Declaramos un lector para el XML indicando el nombre de este
```

```
    StreamReader reader = new StreamReader(fichero, System.Text.Encoding.UTF8);
    //Indicamos cual es el nombre de lector XML a usar en este caso es reader
    XmlTextReader xml = new XmlTextReader(reader);
    xml.Namespaces = false;

    while (xml.Read()) {
      switch (xml.NodeType) {
        case XmlNodeType.Element:
        switch (xml.Name) {
          case "nrepeat":
          numericUpDownRepeat.Value = Convert.ToInt32(xml.ReadString());
          break;
          case "column":
          countrow++;
          break;
          case "Id":
          dataSet1.mitabla.Rows.Add(xml.ReadString());
          //dataSet11.mitabla.Rows[countrow][0] = Convert.ToInt32(xml.ReadString());
          break;
          case "name":
          dataSet1.mitabla.Rows[countrow][1] = xml.ReadString();
          break;
          case "angle":
          dataSet1.mitabla.Rows[countrow][2] = Convert.ToInt32(xml.ReadString());
          break;
          case "speedrot":
          dataSet1.mitabla.Rows[countrow][3] = Convert.ToInt32(xml.ReadString());
          break;
          case "dips":
          dataSet1.mitabla.Rows[countrow][4] = Convert.ToInt32(xml.ReadString());
          break;
          case "hup":
          dataSet1.mitabla.Rows[countrow][5] = Convert.ToDouble(xml.ReadString());
          break;
          case "hdown":
          dataSet1.mitabla.Rows[countrow][6] = Convert.ToDouble(xml.ReadString());
          break;
          case "speedup":
          dataSet1.mitabla.Rows[countrow][7] = Convert.ToDouble(xml.ReadString());
          break;
          case "speeddown":
          dataSet1.mitabla.Rows[countrow][8] = Convert.ToDouble(xml.ReadString());
          break;
          case "tup":
          dataSet1.mitabla.Rows[countrow][9] = Convert.ToInt32(xml.ReadString());
          break;
          case "tdown":
          dataSet1.mitabla.Rows[countrow][10] = Convert.ToInt32(xml.ReadString());
          break;
        }
        break;
      }
    }
    numFilaTotal = countrow + 1;
    xml.Close();
  }

  //Metodo copiado
  private void guardarxml(string fichero) {
    XmlTextWriter xml = new XmlTextWriter(fichero, System.Text.Encoding.UTF8);

    xml.Formatting = Formatting.Indented;
    //Inicializamos el Documento XML
    xml.WriteStartDocument();
    xml.WriteStartElement(null, "Table", null);

    xml.WriteStartElement(null, "tname", null);
    xml.WriteString(fichero);
    xml.WriteEndElement();

    xml.WriteStartElement(null, "date", null);
    xml.WriteString(DateTime.Now.ToString());
```

```
    xml.WriteEndElement();

    xml.WriteStartElement(null, "comment", null);
    xml.WriteString("comentarios varios");
    xml.WriteEndElement();

    xml.WriteStartElement(null, "nrepeat", null);
    xml.WriteString(numericUpDownRepeat.Value.ToString());
    xml.WriteEndElement();

    for (int s = 0; s < dataSet1.mitabla.Rows.Count; s++) {
      //para ver si se corresponde
      //string auxnombre = dataSet11.mitabla.Rows[s]["Name"].ToString();
      xml.WriteStartElement(null, "column", null);

      xml.WriteStartElement(null, "Id", null);
      xml.WriteString(dataSet1.mitabla.Rows[s]["Id"].ToString());
      xml.WriteEndElement();

      xml.WriteStartElement(null, "name", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][1].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "angle", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][2].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "speedrot", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][3].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "dips", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][4].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "hup", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][5].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "hdown", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][6].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "speedup", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][7].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "speeddown", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][8].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "tup", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][9].ToString());
      xml.WriteEndElement();//

      xml.WriteStartElement(null, "tdown", null);
      xml.WriteString(dataSet1.mitabla.Rows[s][10].ToString());
      xml.WriteEndElement();//

      xml.WriteEndElement();

    }
    xml.WriteEndDocument(); //Cerramos el Documento XML
    xml.Close(); //Cerramos el Escritor XML
}

#endregion


 #region Pestaña Row
```

```
private void mostrar(int x) {
   //Ponemos los datos de la fila 'x' en los textBox correspondientes
   putTextBoxAngle(x);
   putTextBoxDips(x);
   putTextBoxHDown(x);
   putTextBoxHUp(x);
   putTextBoxIS(x);
   putTextBoxRS(x);
   putTextBoxTDown(x);
   putTextBoxTUp(x);
   putTextBoxWS(x);
   putLabelRow(x);

   //Marcamos que textbox debe estar activo para añadir nuevos números
   for (int i = 0; i != 9; i++) {
      bTextBoxRow[i] = false;
   }
   bTextBoxRow[0] = true;
   // y lo mostramos mediante un cambio de color
   cambioColor();
}

private void putLabelRow (int i){
   labelRow.Text = "Row: " + i.ToString();
}

private void putLabelRow(string i) {
   labelRow.Text = "Row: " + i;
}

private void clear() {
   //Inicializamos los textBox y label de la pestaña Row
   //para introducir nuevos datos
   putLabelRow("X");
   putTextBoxAngle("");
   putTextBoxDips("");
   putTextBoxHDown("");
   putTextBoxHUp("");
   putTextBoxIS("");
   putTextBoxRS("");
   putTextBoxTDown("");
   putTextBoxTUp("");
   putTextBoxWS("");
   //marcamos que textBox será en el que se introduzcan los datos
   for (int i = 0; i != 9; i++) {
      bTextBoxRow[i] = false;
   }
   bTextBoxRow[0] = true;
   //mediante un cambio de color
   cambioColor();
}

private void cambioColor() {

   //Cambiamos el color de los textBox segun su booleano. Solo 1 blanco y los demas rojo
   if (bTextBoxRow[0]) textBoxAngle.BackColor = System.Drawing.Color.White;
   else textBoxAngle.BackColor = System.Drawing.Color.Red;
   if (bTextBoxRow[1]) textBoxRS.BackColor = System.Drawing.Color.White;
   else textBoxRS.BackColor = System.Drawing.Color.Red;
   if (bTextBoxRow[2]) textBoxDips.BackColor = System.Drawing.Color.White;
   else textBoxDips.BackColor = System.Drawing.Color.Red;
   if (bTextBoxRow[3]) textBoxHUp.BackColor = System.Drawing.Color.White;
   else textBoxHUp.BackColor = System.Drawing.Color.Red;
   if (bTextBoxRow[4]) textBoxHDown.BackColor = System.Drawing.Color.White;
   else textBoxHDown.BackColor = System.Drawing.Color.Red;
   if (bTextBoxRow[5]) textBoxIS.BackColor = System.Drawing.Color.White;
   else textBoxIS.BackColor = System.Drawing.Color.Red;
   if (bTextBoxRow[6]) textBoxWS.BackColor = System.Drawing.Color.White;
   else textBoxWS.BackColor = System.Drawing.Color.Red;
   if (bTextBoxRow[7]) textBoxTUp.BackColor = System.Drawing.Color.White;
   else textBoxTUp.BackColor = System.Drawing.Color.Red;
   if (bTextBoxRow[8]) textBoxTDown.BackColor = System.Drawing.Color.White;
```

```
    else textBoxTDown.BackColor = System.Drawing.Color.Red;
}

public string limites() {
    string sms = "";
    Int32 angle, rs, dips, tup, tdown;
    Double hup, hdown, IS, ws;

    try {
        //Leemos los datos introducidos en los textBox de la pestaña Row
        angle = getAngle();
        rs = getRS();
        dips = getDips();
        tup = getTUp();
        tdown = getTDown();
        hup = getHUp();
        hdown = getHDown();
        IS = getIS();
        ws = getWS();

        //y comparamos con los valores límites.
        if ((angle < 0) || (angle > 360)) sms = sms + "360>=Angle>=0\r\n";
        if ((rs < 0) || (rs > 180)) sms = sms + "180>=RS>= 0\r\n";
        if (dips < 0) sms = sms + "Dips>=0 \r\n";
        if (tup < 0) sms = sms + "TUp>=0\r\n";
        if (tdown < 0) sms = sms + "TDown>=0 \r\n";
        if ((hup < 0) || (hup > 90)) sms = sms + "90>=HUp>=0\r\n";
        if ((hdown < 0) || (hdown > 90)) sms = sms + "90>=HDown>=0\r\n";
        if ((IS < 0) || (IS > 60)) sms = sms + "60>=IS>=0\r\n";
        if ((ws < 0) || (ws > 60)) sms = sms + "60>=WS>=0\r\n";
        return sms;
    }

    //Si hay algun dato que no es un número
    catch (Excepciones ex1) {
        MessageBox.Show("Error. " + ex1.getTexto(), ex1.getTitulo(), System.Windows.Forms.MessageBoxButtons.OK,
        System.Windows.Forms.MessageBoxIcon.Exclamation,
        System.Windows.Forms.MessageBoxDefaultButton.Button1);
        return "Error";
    }
}

#region Botones Row 1

//Boton "1". Añade el caracter 1 al final del textBox
//Comprueba que booleano esta a TRUE del array y añade a ese textbox el valor.
private void button1_Click(object sender, EventArgs e) {
    if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "1";
    else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "1";
    else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "1";
    else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "1";
    else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "1";
    else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "1";
    else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "1";
    else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "1";
    else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "1";
}

//Boton "2". Añade el caracter 2 al final del textBox
private void button2_Click(object sender, EventArgs e) {
    if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "2";
    else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "2";
    else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "2";
    else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "2";
    else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "2";
    else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "2";
    else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "2";
    else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "2";
    else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "2";
}

//Boton "3". Añade el caracter 3 al final del textBox
```

```
private void button3_Click(object sender, EventArgs e) {
   if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "3";
   else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "3";
   else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "3";
   else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "3";
   else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "3";
   else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "3";
   else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "3";
   else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "3";
   else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "3";
}

//Boton "4". Añade el caracter 4 al final del textBox
private void button4_Click(object sender, EventArgs e) {
   if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "4";
   else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "4";
   else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "4";
   else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "4";
   else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "4";
   else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "4";
   else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "4";
   else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "4";
   else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "4";
}

//Boton "5". Añade el caracter 5 al final del textBox
private void button5_Click(object sender, EventArgs e) {
   if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "5";
   else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "5";
   else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "5";
   else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "5";
   else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "5";
   else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "5";
   else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "5";
   else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "5";
   else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "5";

}

//Boton "6". Añade el caracter 6 al final del textBox
private void button6_Click(object sender, EventArgs e) {
   if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "6";
   else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "6";
   else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "6";
   else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "6";
   else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "6";
   else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "6";
   else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "6";
   else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "6";
   else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "6";
}

//Boton "7". Añade el caracter 7 al final del textBox
private void button7_Click(object sender, EventArgs e) {
   if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "7";
   else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "7";
   else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "7";
   else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "7";
   else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "7";
   else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "7";
   else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "7";
   else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "7";
   else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "7";
}

//Boton "8". Añade el caracter 8 al final del textBox
private void button8_Click(object sender, EventArgs e) {
   if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "8";
   else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "8";
   else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "8";
   else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "8";
   else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "8";
```

```
    else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "8";
    else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "8";
    else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "8";
    else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "8";
}

//Boton "9". Añade el caracter 9 al final del textBox
private void button9_Click(object sender, EventArgs e) {
    if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "9";
    else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "9";
    else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "9";
    else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "9";
    else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "9";
    else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "9";
    else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "9";
    else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "9";
    else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "9";
}

//Boton "0". Añade el caracter 0 al final del textBox
private void button10_Click(object sender, EventArgs e) {
    if (bTextBoxRow[0]) textBoxAngle.Text = textBoxAngle.Text + "0";
    else if (bTextBoxRow[1]) textBoxRS.Text = textBoxRS.Text + "0";
    else if (bTextBoxRow[2]) textBoxDips.Text = textBoxDips.Text + "0";
    else if (bTextBoxRow[3]) textBoxHUp.Text = textBoxHUp.Text + "0";
    else if (bTextBoxRow[4]) textBoxHDown.Text = textBoxHDown.Text + "0";
    else if (bTextBoxRow[5]) textBoxIS.Text = textBoxIS.Text + "0";
    else if (bTextBoxRow[6]) textBoxWS.Text = textBoxWS.Text + "0";
    else if (bTextBoxRow[7]) textBoxTUp.Text = textBoxTUp.Text + "0";
    else if (bTextBoxRow[8]) textBoxTDown.Text = textBoxTDown.Text + "0";
}

//Boton "<". Elimina el último caracter del textBox seleccionado
private void button12_Click(object sender, EventArgs e) {
    string sms;
    if (bTextBoxRow[0]) {
        if ((textBoxAngle.Text.Length) != 0){
            sms = textBoxAngle.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxAngle.Text = sms;
        }
    }

    else if (bTextBoxRow[1]) {
        if ((textBoxRS.Text.Length) != 0) {
            sms = textBoxRS.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxRS.Text = sms;
        }
    }

    else if (bTextBoxRow[2]) {
        if ((textBoxDips.Text.Length) != 0){
            sms = textBoxDips.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxDips.Text = sms;
        }
    }

    else if (bTextBoxRow[3]) {
        if ((textBoxHUp.Text.Length) != 0){
            sms = textBoxHUp.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxHUp.Text = sms;
        }
    }

    else if (bTextBoxRow[4]) {
        if ((textBoxHDown.Text.Length) != 0) {
            sms = textBoxHDown.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxHDown.Text = sms;
```

```
        }
    }

    else if (bTextBoxRow[5]) {
        if ((textBoxIS.Text.Length) != 0) {
            sms = textBoxIS.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxIS.Text = sms;
        }
    }

    else if (bTextBoxRow[6]) {
        if ((textBoxWS.Text.Length) != 0) {
            sms = textBoxWS.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxWS.Text = sms;
        }
    }

    else if (bTextBoxRow[7]) {
        if ((textBoxTUp.Text.Length) != 0) {
            sms = textBoxTUp.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxTUp.Text = sms;
        }
    }

    else if (bTextBoxRow[8]) {
        if ((textBoxTDown.Text.Length) != 0){
            sms = textBoxTDown.Text;
            sms = sms.Substring(0, sms.Length - 1);
            textBoxTDown.Text = sms;
        }
    }
    else throw new Excepciones("Calculadora",
                    "No se ha seleccionado ningun sitio donde eliminar el último número",
                    "MenuPrincipal.button12_Click()");

}

#endregion

#region Botones Row 2

//Limpia todos los textBox y pone el cursor en Angle
private void buttonClean_Click(object sender, EventArgs e) {
    clear();
}

//Comprueba y añade, si esta bien los datos, una nueva fila a los procesos
private void buttonAdd_Click(object sender, EventArgs e) {
    string sms = limites();
    try {
        if (System.String.Compare(sms, "") == 0) { //Sino ha salido ningun error.
            //Creamos una nueva fila y añadimos los valores
            DataRow newDR = dataSet1.mitabla.NewRow();
            newDR[2] = getAngle();
            newDR[3] = getRS();
            newDR[4] = getDips();
            newDR[5] = getHUp();
            newDR[6] = getHDown();
            newDR[7] = getIS();
            newDR[8] = getWS();
            newDR[9] = getTUp();
            newDR[10] = getTDown();
            dataSet1.mitabla.Rows.Add(newDR);
            numFilaTotal++;
            clear();
            //Si no se había modificado, se añade el *
            if (guardado) {
                guardado = false;
                putArchivo("*" + getArchivo());
```

```
            }
          }
          else
MessageBox.Show(sms,"Check",MessageBoxButtons.OK,MessageBoxIcon.Exclamation,MessageBoxDefaultButton.Button1);
          numFila = 0;
        }
        catch (Excepciones ex1) {
          MessageBox.Show("Error. " + ex1.getTexto(), ex1.getTitulo(), System.Windows.Forms.MessageBoxButtons.OK,
          System.Windows.Forms.MessageBoxIcon.Exclamation,
          System.Windows.Forms.MessageBoxDefaultButton.Button1);
        }
      }

      //Segun la fila seleccionada, modifica los valores por los puestos en los textBox
      private void buttonModify_Click(object sender, EventArgs e) {
        Int32 angle, rs, dips, tup, tdown;
        Double hup, hdown, IS, ws;
        string sms = "";

        try {
          //Leemos los datos de los textBox
          angle = getAngle();
          rs = getRS();
          dips = getDips();
          tup = getTUp();
          tdown = getTDown();
          hup = getHUp();
          hdown = getHDown();
          IS = getIS();
          ws = getWS();

          //Comprobamos si estan entre los límites
          if (System.String.Compare(limites(), "") == 0) { //Sino ha salido ningun error.
            if (getDataAngle(numFila) != angle) sms = sms + "Angle ";
            if (getDataDips(numFila) != dips) sms = sms + "Dips ";
            if (getDataHDown(numFila) != hdown) sms = sms + "HDown ";
            if (getDataHUp(numFila) != hup) sms = sms + "HUp ";
            if (getDataIS(numFila) != IS) sms = sms + "IS ";
            if (getDataRS(numFila) != getRS()) sms = sms + "RS ";
            if (getDataTDown(numFila) != tdown) sms = sms + "TDown ";
            if (getDataTUp(numFila) != tup) sms = sms + "TUp ";
            if (getDataWS(numFila) != ws) sms = sms + "WS ";

            //Comprobamos si se han cambiado algún dato, y si hay alguno, se modifican
            if (String.Compare(sms, "") != 0) {
              if (MessageBox.Show("Do you want to change the values " + sms + " on the row number " + numFila.ToString()
                , "Attention", MessageBoxButtons.OKCancel, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) ==
DialogResult.OK) {
                putDataAngle(numFila, angle);
                putDataDips(numFila, dips);
                putDataHDown(numFila, hdown);
                putDataHUp(numFila, hup);
                putDataIS(numFila, IS);
                putDataRS(numFila, getRS());
                putDataTDown(numFila, tdown);
                putDataTUp(numFila, tup);
                putDataWS(numFila, ws);
                clear();
                //Si no se había modificado, se añade el *
                if (guardado) {
                  guardado = false;
                  putArchivo("*" + getArchivo());
                }
              }
              else mostrar(numFila);
            }
            else MessageBox.Show("No data has changed", "Attention",
              MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
          }
          else MessageBox.Show(sms, "Check", MessageBoxButtons.OK, MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
          numFila = 0;
```

```
    }
    catch (Excepciones ex1) {
      MessageBox.Show("Error. " + ex1.getTexto() ,ex1.getTitulo() , System.Windows.Forms.MessageBoxButtons.OK,
      System.Windows.Forms.MessageBoxIcon.Exclamation,
      System.Windows.Forms.MessageBoxDefaultButton.Button1);
    }
}

//Boton "Erase". Borra la fila que se muestra en la pestaña row.
private void buttonErase_Click(object sender, EventArgs e) {
    int fila = numFila;
    string sms = "Do you want to erase the row number " + fila.ToString();

    if (fila != -1) {
      if (MessageBox.Show(sms, "Attention", MessageBoxButtons.OKCancel,
            MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) == DialogResult.OK) {
        dataSet1.mitabla.Rows[fila].Delete();
        numFilaTotal--;
        clear();
        numFila = 0;
        //Si no se había modificado, se añade el *
        if (guardado) {
          guardado = false;
          putArchivo("*" + getArchivo());
        }
      }
    }
    else MessageBox.Show("Error al borrado");
}

//Boton "Next". Muestra la fila siguiente. Si es la última fila, muestra la primera.
private void buttonNextButton_Click(object sender, EventArgs e) {
    //Comprobamos que estan entre los límites para mostrar
    if (dataSet1.mitabla.Count != 0){
      if (numFila != numFilaTotal - 1) {
        numFila++;
        mostrar(numFila);
      }
      else if (numFila == numFilaTotal - 1) {
        numFila = 0;
        mostrar(numFila);
      }
    }
}

//Boton "Before". Muestra la fila anterior. Si es la primera fila, muestra la última introducida.
private void buttonBeforeButton_Click(object sender, EventArgs e) {
    //Comprobamos que esta entre los límites para mostrar.
    if (dataSet1.mitabla.Count != 0){
      if (numFila != 0) {
        numFila--;
        mostrar(numFila);
      }
      else {
        numFila = numFilaTotal - 1;
        mostrar(numFila);
      }
    }
}

#endregion

#region Eventos textBoxXX

//Selecciona el textbox Angle para introducir los datos.
private void textBoxAngle_GotFocus(object sender, EventArgs e) {
    //Limpiamos el textBox y ponemos el booleano en el array a true
    putTextBoxAngle("");
     for (int i = 0; i != 9; i++) {
      bTextBoxRow[i] = false;
    }
    //Para introducir en ese textbox los datos y cambiar el color
```

```
    bTextBoxRow[0] = true;
    cambioColor();
}

//Selecciona el textbox RS para introducir los datos.
private void textBoxRS_GotFocus(object sender, EventArgs e) {
    putTextBoxRS("");
    for (int i = 0; i != 9; i++) {
        bTextBoxRow[i] = false;
    }
    bTextBoxRow[1] = true;
    cambioColor();
}

//Selecciona el textbox Dips para introducir los datos.
private void textBoxDips_GotFocus(object sender, EventArgs e) {
    putTextBoxDips("");
    for (int i = 0; i != 9; i++) {
        bTextBoxRow[i] = false;
    }
    bTextBoxRow[2] = true;
    cambioColor();
}

//Selecciona el textbox HUp para introducir los datos.
private void textBoxHUp_GotFocus(object sender, EventArgs e) {
    putTextBoxHUp("");
    for (int i = 0; i != 9; i++) {
        bTextBoxRow[i] = false;
    }
    bTextBoxRow[3] = true;
    cambioColor();
}

//Selecciona el textbox HDown para introducir los datos.
private void textBoxHDown_GotFocus(object sender, EventArgs e) {
    putTextBoxHDown("");
    for (int i = 0; i != 9; i++) {
        bTextBoxRow[i] = false;
    }
    bTextBoxRow[4] = true;
    cambioColor();
}

//Selecciona el textbox IS para introducir los datos.
private void textBoxIS_GotFocus(object sender, EventArgs e) {
    putTextBoxIS("");
    for (int i = 0; i != 9; i++) {
        bTextBoxRow[i] = false;
    }
    bTextBoxRow[5] = true;
    cambioColor();
}

//Selecciona el textbox WS para introducir los datos.
private void textBoxWS_GotFocus(object sender, EventArgs e) {
    putTextBoxWS("");
    for (int i = 0; i != 9; i++) {
        bTextBoxRow[i] = false;
    }
    bTextBoxRow[6] = true;
    cambioColor();
}

//Selecciona el textbox TUp para introducir los datos.
private void textBoxTUp_GotFocus(object sender, EventArgs e) {
    putTextBoxTUp("");
    for (int i = 0; i != 9; i++) {
        bTextBoxRow[i] = false;
    }
    bTextBoxRow[7] = true;
    cambioColor();
```

```
}

//Selecciona el textbox TDown para introducir los datos.
private void textBoxTDown_GotFocus(object sender, EventArgs e) {
   putTextBoxTDown("");
   for (int i = 0; i != 9; i++) {
      bTextBoxRow[i] = false;
   }
   bTextBoxRow[8] = true;
   cambioColor();
}

#endregion

#region Metodos para escribir en los textBox

public void putTextBoxAngle(string sms) {
   textBoxAngle.Text = sms;
}

public void putTextBoxRS(string sms) {
   textBoxRS.Text = sms;
}

public void putTextBoxDips(string sms) {
   textBoxDips.Text = sms;
}

public void putTextBoxHUp(string sms) {
   textBoxHUp.Text = sms;
}

public void putTextBoxHDown(string sms) {
   textBoxHDown.Text = sms;
}

public void putTextBoxIS(string sms) {
   textBoxIS.Text = sms;
}

public void putTextBoxWS(string sms) {
   textBoxWS.Text = sms;
}

public void putTextBoxTUp(string sms) {
   textBoxTUp.Text = sms;
}

public void putTextBoxTDown(string sms) {
   textBoxTDown.Text = sms;
}

private void putTextBoxAngle(int i) {
   textBoxAngle.Text = dataSet1.mitabla[i]._Angle__º_.ToString();
}

private void putTextBoxRS(int i) {
   textBoxRS.Text = dataSet1.mitabla[i]._Rotation_speed__º_s_.ToString();
}

private void putTextBoxDips(int i) {
   textBoxDips.Text = dataSet1.mitabla[i].Dips.ToString();
}

private void putTextBoxHUp(int i) {
   textBoxHUp.Text = dataSet1.mitabla[i]._H_Up__mm_.ToString();
}

private void putTextBoxHDown(int i) {
   textBoxHDown.Text = dataSet1.mitabla[i]._H_Down__mm_.ToString();
}
```

```
private void putTextBoxIS(int i) {
   textBoxIS.Text = dataSet1.mitabla[i]._Immersion_speed__mm_s_.ToString();
}

private void putTextBoxWS(int i) {
   textBoxWS.Text = dataSet1.mitabla[i]._Withdrawal_speed__mm_s_.ToString();
}

private void putTextBoxTUp(int i) {
   textBoxTUp.Text = dataSet1.mitabla[i].T_Up.ToString();
}

private void putTextBoxTDown(int i) {
   textBoxTDown.Text = dataSet1.mitabla[i].T_Down.ToString();
}

#endregion

#region Metodos para obtener los Strings de los textBox

public string getTextBoxAngle() {
   //Leemos los datos del textBox sini espacios
   return textBoxAngle.Text.Trim();
}

public string getTextBoxRS() {
   return textBoxRS.Text.Trim();
}

public string getTextBoxDips() {
   return textBoxDips.Text.Trim();
}

public string getTextBoxTUp() {
   return textBoxTUp.Text.Trim();
}

public string getTextBoxTDown() {
   return textBoxTDown.Text.Trim();
}

public string getTextBoxHUp() {
   return textBoxHUp.Text.Trim();
}

public string getTextBoxHDown() {
   return textBoxHDown.Text.Trim();
}

public string getTextBoxIS() {
   return textBoxIS.Text.Trim();
}

public string getTextBoxWS() {
   return textBoxWS.Text.Trim();
}

#endregion

/*Convierte los strings de los textbox a los datos del data set. Si en algun caso no se ha introducido numeros,
o no son correctos los caracteres, devuelve un valor -1 como error. */
#region String a los valores del Data Set

private Int32 getAngle() {
   Int32 i32;
   try {
      //Convertimos el texto al tipo que se necesita para introducir en la tabla
      i32 = System.Convert.ToInt32(getTextBoxAngle());
      return i32;
   }
   catch (Exception) {
      //Si se produce un error se devuelve -1 (fuera del límite)
```

```
        return -1;
    }
}

private Int32 getRS() {
    Int32 i32;
    try {
        i32 = System.Convert.ToInt32(getTextBoxRS());
        return i32;
    }
    catch (Exception) {
        return -1;
    }
}

private Int32 getDips() {
    Int32 i32;
    try {
        i32 = System.Convert.ToInt32(getTextBoxDips());
        return i32;
    }
    catch (Exception) {
        return -1;
    }
}

private Int32 getTUp() {
    Int32 i32;
    try {
        i32 = System.Convert.ToInt32(getTextBoxTUp());
        return i32;
    }
    catch (Exception) {
        return -1;
    }
}

private Int32 getTDown() {
    Int32 i32;
    try {
        i32 = System.Convert.ToInt32(getTextBoxTDown());
        return i32;
    }
    catch (Exception) {
        return -1;
    }
}

private Double getHUp() {
    Double d;
    try {
        d = System.Convert.ToDouble(getTextBoxHUp());
        return d;
    }
    catch (Exception) {
        return -1;
    }
}

private Double getHDown() {
    Double d;
    try {
        d = System.Convert.ToDouble(getTextBoxHDown());
        return d;
    }
    catch (Exception) {
        return -1;
    }
}

private Double getIS() {
    Double d;
```

```
    try {
      d = System.Convert.ToDouble(getTextBoxIS());
      return d;
    }
    catch (Exception) {
      return -1;
    }
  }

  private Double getWS() {
    Double d;
    try {
      d = System.Convert.ToDouble(getTextBoxWS());
      return d;
    }
    catch (Exception) {
      return -1;
    }
  }

  #endregion

  #endregion

  //Hasta aqui comentado los metodos en general y que hace cada uno.

  #region Pestaña Device

  public void putLabelProcess(string sms) {
    labelProcess.Text = sms;
  }

  public void putLabelPosx(string sms) {
    labelPosx.Text = sms;
  }

  public void putLabelPosy(string sms) {
    labelPosy.Text = sms;
  }

  public void putLabelStatus(string sms) {
    labelStatus.Text = sms;
  }

  public void putLabelCiclo(string sms) {
    labelCiclo.Text = sms;
  }

  public void putLabelTotalT(string sms){
    labelTotalT.Text = sms;
  }

  public void putLabelPT(string sms){
    labelPt.Text = sms;
  }

  private void buttonRun_Click(object sender, EventArgs e) {
    AccionButtonRun();
  }

  private void buttonStop_Click(object sender, EventArgs e) {
    AccionButtonStop();
  }

  private void buttonReset_Click(object sender, EventArgs e) {
    AccionButtonReset();
  }

  //Copiado el día 25-10
  private void AccionButtonRun() {
    if (buttonRun.Text == "Run") {
      if (RevisarDatos() == true) {
```

```
            repeatotal = Convert.ToInt32(numericUpDownRepeat.Value);
            string auxtiempo = Calctiempototal();
            //variables a cero
            numdip = 0; inicio = 0; iniciclo = true; fase = 0; numrepeat = 0;

            if (pauseprog == true) pauseprog = false;
            timerConocer.Enabled = true;
            timerprogram.Enabled = true;
            //timerreloj.Enabled = true;
            timerposicion.Enabled = true;
            buttonRun.Text = "Pause";
            irun = false; ipause = true; istop = true;
            limpiarTextBoxDevice();
            addTextBoxDevice("Total process time estimated: \r\n" + auxtiempo + "\r\n");
            addTextBoxDevice("   START\r\n");
            //statusStrip1.Visible = true;
            //statusStrip1.Enabled = true;

            //richTextBox1.SelectionStart = richTextBox1.Text.Length;
            //richTextBox1.ScrollToCaret();
            //richTextBox1.Refresh();

            //buttonupdatedb.Enabled = false;
          }
        }
        else if (buttonRun.Text == "Pause") {
          //timerprogram.Enabled = false;
          pauseprog = true;
          addTextBoxDevice("   PAUSE\n\r");
          buttonRun.Text = "Restart";
          irun = true; ipause = false; istop = true;
          //buttonreset.Enabled = true;
        }
        else {
          //if (timerprogram.Enabled == false) timerprogram.Enabled = true;
          pauseprog = false;
          if (timerposicion.Enabled == false) timerposicion.Enabled = true;
          //statusStrip1.Visible = true;
          //statusStrip1.Enabled = true;
          addTextBoxDevice("   RESTART\n\r");
          irun = false; ipause = true; istop = true;
          buttonRun.Text = "Pause";
        }
      }

      //Copiado el día 25-10
      private void AccionButtonReset() {
        try {
          if (timerprogram.Enabled == true) addTextBoxDevice("RESET\n\r");
          timerprogram.Enabled = false;
          timerposicion.Enabled = false;
          //statusStrip1.Visible = false;

          Reset();
          iniciclo = true; inicio = 0; numrepeat = 0; fase = 0; numdip = 0;
          hpos = 0; hpos_anterior = 0; hposy = 0; hpos_anteriory = 0;
          buttonRun.Text = "RUN";

          //richTextBox1.Text += "RESET\n";
          //buttonRun.Enabled = false;
          imov = false;
          //buttonupdatedb.Enabled = true;
        }
        catch {
          MessageBox.Show("Needs to close dipping window");
        }
      }

      private void AccionButtonStop() {
        timerConocer.Enabled = false;
        timerRecibir.Enabled = false;
        timerposicion.Enabled = false;
```

```
    timerprogram.Enabled = false;
    timerreloj.Enabled = false;
    int periodo = 25 / 3;//25 / 4;//25/ (5 mm/s)
    //char funcion = 'ù'; //0x96 Creo que es este.
    Byte[] OUTBuffer = new Byte[65];

    OUTBuffer[0] = 0x2D;
    OUTBuffer[1] = 0x96;
    OUTBuffer[2] = Convert.ToByte(((periodo & 0xFF0000) >> 16));
    OUTBuffer[3] = Convert.ToByte(((periodo & 0xFF00) >> 8));
    OUTBuffer[4] = Convert.ToByte((periodo & 0x00FF));
    OUTBuffer[5] = checksum(OUTBuffer, 2, 5);
    OUTBuffer[6] = 0x23;

    try {
      conexion.enviarDato(OUTBuffer,7);
    }
    catch (Exception) {
      MessageBox.Show("Error en AccionButtonStop()");
    }
    addTextBoxDevice("\r\n---STOP---\r\n");
}

public string getTextBoxDevice() {
    return textBoxDevice.Text;
}

public void addTextBoxDevice(string sms) {
    if (textBoxDevice.InvokeRequired) {
      SetTextCallback d = new SetTextCallback(addTextBoxDevice);
      this.Invoke(d, sms);
    }
    else {
      textBoxDevice.Text = textBoxDevice.Text + sms;
      textBoxDevice.SelectionStart = textBoxDevice.TextLength;
      textBoxDevice.ScrollToCaret();
    }
}

public void limpiarTextBoxDevice() {
    textBoxDevice.Text = "";
}

#endregion



#region DataSet1


public int getNumberOfRow() {
    return dataSet1.mitabla.Rows.Count;
}

public DataRow getDataRow(int fila) {
    return dataSet1.mitabla[fila];
}

#region Escribe el valor deseado en la fila i
private void putDataAngle(int i, Int32 i32) {
    dataSet1.mitabla[i][2]= i32;
}

private void putDataRS(int i, Int32 i32) {
    dataSet1.mitabla[i][3] = i32;
}

private void putDataDips(int i, Int32 i32) {
    dataSet1.mitabla[i][4] = i32;
}

private void putDataHUp(int i, Double d) {
```

```
    dataSet1.mitabla[i][5] = d;
}

private void putDataHDown(int i, Double d) {
    dataSet1.mitabla[i][6] = d;
}

private void putDataIS(int i, Double d) {
    dataSet1.mitabla[i][7] = d;
}

private void putDataWS(int i, Double d) {
    dataSet1.mitabla[i][8] = d;
}

private void putDataTUp(int i, Int32 i32) {
    dataSet1.mitabla[i][9] = i32;
}

private void putDataTDown(int i, Int32 i32) {
    dataSet1.mitabla[i][10] = i32;
}
#endregion

#region Devuelve el valor de la fila i del dato deseado
private Int32 getDataAngle(int i) {
    return Convert.ToInt32(dataSet1.mitabla.Rows[i][2]);
}

private Int32 getDataRS(int i) {
    return Convert.ToInt32(dataSet1.mitabla.Rows[i][3]);
}

private Int32 getDataDips(int i) {
    return Convert.ToInt32(dataSet1.mitabla.Rows[i][4]);
}

private Double getDataHUp(int i) {
    return Convert.ToDouble(dataSet1.mitabla.Rows[i][5]);
}

private Double getDataHDown(int i) {
    return Convert.ToDouble(dataSet1.mitabla.Rows[i][6]);
}

private Double getDataIS(int i) {
    return Convert.ToDouble(dataSet1.mitabla.Rows[i][7]);
}

private Double getDataWS(int i) {
    return Convert.ToDouble(dataSet1.mitabla.Rows[i][8]);
}

private Int32 getDataTUp(int i) {
    return Convert.ToInt32(dataSet1.mitabla.Rows[i][9]);
}

private Int32 getDataTDown(int i) {
    return Convert.ToInt32(dataSet1.mitabla.Rows[i][10]);
}
#endregion

#endregion



#region Temporizadores y sus acciones

#region Siempre activados Conocer - Recibir. Thread del menu principal del DipDR dividido en dos partes

private void timerConocer_Tick(object sender, EventArgs e) {
    timerConocer.Enabled = false;
```

```
    timerRecibir.Enabled = true;
    AccionConocer();
}

private void AccionConocer() {
    Byte[] OUTBuffer = new Byte[65];
    OUTBuffer[0] = 0x2D;
    if (envio == 0) {
        OUTBuffer[1] = 0x37;
        envio = 1;
    }
    else if (envio == 1) {
        OUTBuffer[1] = 0x81;
        envio = 0;
    }
    OUTBuffer[2] = 0x23;

    try {
        enviar(OUTBuffer, 3);
    }
    catch (Exception) {
        MessageBox.Show("MenuPrincipalMini.cs AccionConocer()");
    }
}

private void timerRecibir_Tick(object sender, EventArgs e) {
    timerConocer.Enabled = true;
    timerRecibir.Enabled = false;
    AccionRecibir();
}

private void AccionRecibir() {
    Byte[] INBuffer = new Byte[65];
    Byte check;
    try {
        INBuffer = conexion.recibirDato();
        if (INBuffer != null) {
            if (INBuffer[1] == 0x37) {
                check = checksum(INBuffer, 2, 8);
                if (check == INBuffer[8]){
                    //MessageBox.Show("OK 0x37");
                    pasosdevy = Convert.ToUInt32((INBuffer[4] << 16)) +
                            Convert.ToUInt32((INBuffer[3] << 8)) +
                            INBuffer[2];
                    pasosdev = Convert.ToUInt32((INBuffer[7] << 16)) +
                           Convert.ToUInt32((INBuffer[6] << 8)) +
                           INBuffer[5];
                }
            }
            else if (INBuffer[1] == 0x81) {
                check = checksum(INBuffer, 2, 6);
                if (check == INBuffer[6]) {
                    //MessageBox.Show("OK 0x81");
                    if ((INBuffer[1] == 0x81) && (INBuffer[2] >= 0x01)) {
                        //PushbuttonPressed = false;
                        posunknown = true;
                    }
                    if ((INBuffer[1] == 0x81) && (INBuffer[2] == 0x00)) {
                        //PushbuttonPressed = true;
                        posunknown = false;
                        //primlect = false;
                    }
                    if ((INBuffer[1] == 0x81) && (INBuffer[3] >= 0x01)) {
                        Is_Busyx = true;
                    }
                    else if ((INBuffer[1] == 0x81) && (INBuffer[3] == 0x00)) {
                        Is_Busyx = false;
                    }
                    if ((INBuffer[1] == 0x81) && (INBuffer[4] >= 0x01)) {
                        Is_Busyy = true;
                    }
                    else if ((INBuffer[1] == 0x81) && (INBuffer[4] == 0x00)) {
```

```
                    Is_Busyy = false;
                }
                if ((INBuffer[1] == 0x81) && (INBuffer[5] >= 0x01)) {
                    Is_Reset = true;
                }
                else if ((INBuffer[1] == 0x81) && (INBuffer[5] == 0x00)) {
                    Is_Reset = false;
                }
            }
        }
      }
      //else MessageBox.Show("NULL");
    }
    catch (Exception) {
      MessageBox.Show("MenuPrincipalMini.cs AccionRecibir()");
    }
  }

  #endregion

  #region Bucle 2 Program - Reloj - Posición
  private void timerprogram_Tick(object sender, EventArgs e) {
    timerprogram.Enabled = false;
    timerreloj.Enabled = true;
    AccionProgram();
  }

  private void timerreloj_Tick(object sender, EventArgs e) {
    timerreloj.Enabled = false;
    timerposicion.Enabled = true;
    AccionReloj();
  }

  private void timerposicion_Tick(object sender, EventArgs e) {
    timerposicion.Enabled = false;
    timerprogram.Enabled = true;
    AccionPosicion();
  }

  private void AccionPosicion() {
    if (Is_Busyx == false && Is_Busyy == false) {

      if (hpos != hpos_anterior) {
        if (hpos <= 51200)   //limite de las guias
        {
          if (hpos > hpos_anterior) Girar(hpos - hpos_anterior, 0, periodox);
          else Girar(hpos_anterior - hpos, 1, periodox);
          if (Convert.ToInt32(hpos) - Convert.ToInt32(hpos_anterior) > 0) {
            if (tiempoproctot < (periodox * (hpos - hpos_anterior)) / 100000) tiempoproctot = (periodox * 2 * (hpos -
hpos_anterior)) / 100000;  //cada 100ms
          }
          else {
            if (tiempoproctot < (periodox * (hpos - hpos_anterior)) / 100000) tiempoproctot = (periodox * 2 * (hpos_anterior -
hpos)) / 100000;  //cada 100ms
          }
          boolproceso = true;
          tiempopropar = 0;

        }
        else MessageBox.Show("Out of X Axis Range. Modify Posx: Row - " + fase + 1);
      }
      if (hposy != hpos_anteriory) {
        if (hposy <= 236790)   //limite de las guias
        {
          if (hposy > hpos_anteriory) Sub(hposy - hpos_anteriory, 0, periodoy);
          else Sub(hpos_anteriory - hposy, 1, periodoy);
          if (Convert.ToInt32(hposy) - Convert.ToInt32(hpos_anteriory) > 0) {
            if (tiempoproctot < (periodoy * (hposy - hpos_anteriory)) / 100000) tiempoproctot = (periodoy * (hposy -
hpos_anteriory)) / 100000;  //cada 100ms
          }
          else {
```

```
                if (tiempoproctot < (periodoy * (hpos_anteriory - hposy)) / 100000) tiempoproctot = (periodoy * (hpos_anteriory -
hposy)) / 100000;  //cada 100ms
                }
                boolproceso = true;
                tiempopropar = 0;
              }
              else MessageBox.Show("Out of Y Axis Range. Modify Posy: Row - " + fase + 1);
            }

            if (tmovimiento == true) {
              tmovimiento = false;
              if (inicio == 1) inicio = 2;
              else if (inicio == 2) inicio = 3;
              //else if (inicio == 3) inicio = 4;
              else if (inicio == 4) inicio = 5;
              else if (inicio == 6) inicio = 7;
            }

            if (boolproceso == true) {
              //labeltoperacion.Text = "Tiempo de operacion : " + tiempopropar.ToString() + " / " + tiempoproctot.ToString();

              if (timerreloj.Enabled == false && inicio == 4) putLabelPT((tiempopropar / 10).ToString() + "/" + (tiempoproctot /
10).ToString());
              if (tiempopropar < tiempoproctot) tiempopropar++;
              else if (tiempopropar >= tiempoproctot) {
                boolproceso = false;
                tiempoproctot = 0;
              }
            }
            hpos_anterior = hpos;
            hpos_anteriory = hposy;

          }
          //muestra el tiempo de operacion
          if (boolproceso == true) {
            //labeltoperacion.Text = "Tiempo de operacion : " + tiempopropar.ToString() + " / " + tiempoproctot.ToString();
            if (timerreloj.Enabled == false) putLabelPT((tiempopropar / 10).ToString() + "/" + (tiempoproctot / 10).ToString());
            if (tiempopropar < tiempoproctot) tiempopropar++;
            else if (tiempopropar >= tiempoproctot || timerreloj.Enabled == false) {
              boolproceso = false;
              tiempoproctot = 0;
            }
          }

          //muestra la posicion
          if (first == true) //si es la priemra vez muestra lo que esta guardado en el micro
          {
            first = false;
            hpos = Convert.ToUInt32(pasosdev); hpos_anterior = Convert.ToUInt32(pasosdev);  //* 6 / 160Inicia con ese numero de pasos
guardados
            hposy = Convert.ToUInt32(pasosdevy); hpos_anteriory = Convert.ToUInt32(pasosdevy);
          }
          if (timerreloj.Enabled == false) {
            progressBar1.Maximum = (int)tiempoproctot;
            if (tiempopropar < progressBar1.Maximum && tiempopropar > progressBar1.Minimum) progressBar1.Value =
(int)tiempopropar;
          }
          putLabelPosx(Convert.ToInt32((pasosdev / avancepasgiro)).ToString() + " º");
          putLabelPosy(Convert.ToInt32((pasosdevy / avancepasalt)).ToString() + " mm");

          //para detectar cambios en el menu de MenuInicial
          if (asaveas == true) {
            asaveas = false;
            //GuardarComo();
          }
          if (asave == true) {
            asaveas = false;
            //Guardar();
          }
          if (acreset == true) {
            acreset = false;
            AccionButtonReset();
```

```
    }
    if (acrun == true) {
      acrun = false;
      AccionButtonRun();
    }
    if (acpause == true) {
      acpause = false;
      AccionButtonRun();
    }
  }

  private void AccionProgram() {
    if (Is_Busyx == false && Is_Busyy == false && pauseprog == false) {
      if (numrepeat < repeatotal) { //Contador de repeticiones
        if (iniciclo == true) {
          putLabelCiclo("Repeat: " + (numrepeat + 1).ToString());
          iniciclo = false;
        }

        if (fase < dataSet1.mitabla.Rows.Count && boolproceso == false) { //Contador de filas del ciclo
          if (inicio == 0) {
            addTextBoxDevice((dataSet1.mitabla[fase][1]).ToString() + " -> " + DateTime.Now.ToString() + "\r\n");
            inicio++;
          }

          if (inicio == 1) {
            //Do Row
            angle = Convert.ToInt32(Convert.ToDouble(dataSet1.mitabla[fase][2]) * avancepasgiro);
            posh1 = Convert.ToInt32(Convert.ToDouble(dataSet1.mitabla[fase][5]) * avancepasalt);
            posh2 = Convert.ToInt32(Convert.ToDouble(dataSet1.mitabla[fase][6]) * avancepasalt);
            speedgiro = Convert.ToInt32(cvelgiro / (Convert.ToDouble(this.dataSet1.mitabla[fase][3])));
            speedsub = Convert.ToInt32(cvelalt / (Convert.ToDouble((double)this.dataSet1.mitabla[fase][7])));
            time1 = (int)this.dataSet1.mitabla[fase][9];
            time2 = (int)this.dataSet1.mitabla[fase][10];
            //labelProcess.Text = (dataSet1.mitabla[fase][1]).ToString();
            putLabelStatus(" - ");

            if (hposy > posh1) {
              periodoy = Convert.ToUInt32(speedsub);
              hposy = Convert.ToUInt32(posh1);
              tmovimiento = true;
            }
            else inicio++;
          }
          if (inicio == 2) {
            periodox = Convert.ToUInt32(speedgiro);
            periodoy = Convert.ToUInt32(speedsub);
            hpos = Convert.ToUInt32(angle);
            //hposy = Convert.ToUInt32(posy);
            tmovimiento = true;
          }
          if (inicio == 3)   //Posicion de altura 2 antes de hacer los dippings
            {
            //hposy = Convert.ToUInt32(posh1);
            //periodoy = Convert.ToUInt32(speedsub);
            //tmovimiento = true;
            inicio++;
          }
          if (numdip < (int)dataSet1.mitabla[fase][4]) {
            if (inicio == 4) {  //Posicion de altura 2 antes de hacer los dippings
              putLabelStatus("Dips: " + (numdip + 1).ToString() + " / " + (dataSet1.mitabla[fase][4]).ToString());
              hposy = Convert.ToUInt32(posh2);
              periodoy = Convert.ToUInt32(speedsub);
              tmovimiento = true;
            }

            if (inicio == 5) {
              if (time2 != 0) {
                if (timerreloj.Enabled == false) timerreloj.Enabled = true;
              }
              else inicio++;
              if (timespera >= time2 && time2 != 0) {
```

```
                    inicio++;
                    timerreloj.Enabled = false;
                    timespera = 0;
                  }

                }

                if (inicio == 6) {   //Posicion de altura 2 antes de hacer los dippings
                  hposy = Convert.ToUInt32(posh1);
                  periodoy = Convert.ToUInt32(speedsub);
                  tmovimiento = true;
                }

                if (inicio == 7) {
                  if (time1 != 0) {
                    if (timerreloj.Enabled == false) timerreloj.Enabled = true;
                  }
                  else {
                    inicio = 4;
                    numdip++;
                  }
                  if (timespera >= time1 && time1 != 0) {
                    inicio = 4;
                    numdip++;
                    timerreloj.Enabled = false;
                    timespera = 0;
                  }

                }
              }
              else {
                inicio = 0;
                fase++;
                numdip = 0;
              }

            }
            else if (fase >= dataSet1.mitabla.Rows.Count) {
              timerreloj.Enabled = false;
              iniciclo = true;
              numrepeat++;
              fase = 0;
              inicio = 0;
            }
          }
          else {
            timerprogram.Enabled = false;
            timerConocer.Enabled = false;
            timerRecibir.Enabled = false;
            timerposicion.Enabled = false;
            timerreloj.Enabled = false;
            numrepeat = 0;
            addTextBoxDevice("\r\n---END---\r\n");
            buttonRun.Text = "Run";
            buttonReset.Enabled = true;
            ireset = true;
          }
        }
      }

      private void AccionReloj() {
        timespera++;
        if (inicio == 5 && timespera <= time2) {
          putLabelPT((timespera).ToString() + "/" + (time2).ToString());
          //progressBar1.Maximum = (int)time2;
          //if (timespera < progressBar1.Maximum && timespera > progressBar1.Minimum) progressBar1.Value = (int)timespera;
        }
        if (inicio == 7 && timespera <= time1) {
          putLabelPT((timespera).ToString() + "/" + (time1).ToString());
          //progressBar1.Maximum = (int)time1;
          //if (timespera < progressBar1.Maximum && timespera > progressBar1.Minimum) progressBar1.Value = (int)timespera;
        }
```

```
}

#endregion

//Comprobación
private void timerReenviar_Tick(object sender, EventArgs e) {
   enviar(send, isend);
}

private void reenviar(Byte[] b, int i) {
   for (int j = 0; j < i; j++) send[j] = b[j];
   isend = i;
}

private void enviar(Byte[] a, int b) {
   try {
      while (conexion.IsOcupado()) ;
      conexion.ocupado();
      conexion.enviarDato(a, b);
      conexion.liberar();
      timerConocer.Enabled = false;
      timerRecibir.Enabled = false;
   }
   catch (Exception) {
      MessageBox.Show("MenuPrincipalMini.cs enviar()");
   }
}

private bool recibir(){
   Byte[] INBuffer = conexion.recibirDato();
   if ((INBuffer[1] == 0x4F) && (INBuffer[2] == 0x6B)) return true;
   else return false;
}

//Array de bytes, inicio y fin de lo que se quiere calcular.
private Byte checksum(Byte[] c, int a, int b) {
   int sumacheck = 0;
   for (int j = a; j < b; j++) sumacheck += c[j];
   return Convert.ToByte((sumacheck) & 0x00FF);
}

#endregion



#region Subir, Girar y Reset

public void Sub(long steps, int dir, uint periodo) {

   Byte[] OUTBuffer = new byte[65];
   int i;
   timerConocer.Enabled = false;
   timerRecibir.Enabled = false;
   OUTBuffer[0] = 0x2D;
   if (periodo < 366) {// && motor == 2)
      int contimer = Convert.ToUInt16(steps / 0xFFF);
      if (steps % 0xFFF != 0) contimer++;
      int division = Convert.ToUInt16(steps / contimer);
      int resto = Convert.ToUInt16(steps % contimer); //Dara en la ultima division los pasos restantes

      OUTBuffer[1] = 0x85; //altura
      OUTBuffer[2] = Convert.ToByte(contimer);
      OUTBuffer[3] = Convert.ToByte(((division & 0xFF00) >> 8));
      OUTBuffer[4] = Convert.ToByte((division & 0x00FF));
      OUTBuffer[5] = Convert.ToByte(resto);
      OUTBuffer[6] = Convert.ToByte(dir);
      OUTBuffer[7] = Convert.ToByte(((periodo & 0xFF0000) >> 16));
      OUTBuffer[8] = Convert.ToByte(((periodo & 0xFF00) >> 8));
      OUTBuffer[9] = Convert.ToByte((periodo & 0x00FF));
      OUTBuffer[10] = checksum(OUTBuffer,2, 10);
      OUTBuffer[11] = 0x23;
```

```
            reenviar(OUTBuffer, 12);
        }
        else {
            OUTBuffer[1] = 0x90; //altura
            OUTBuffer[2] = Convert.ToByte(((steps & 0xFF0000) >> 16));
            OUTBuffer[3] = Convert.ToByte(((steps & 0xFF00) >> 8));
            OUTBuffer[4] = Convert.ToByte((steps & 0x00FF));
            OUTBuffer[5] = Convert.ToByte(dir);
            OUTBuffer[6] = Convert.ToByte(((periodo & 0xFF00) >> 8));
            OUTBuffer[7] = Convert.ToByte((periodo & 0x00FF));
            OUTBuffer[8] = checksum(OUTBuffer,2, 8);
            OUTBuffer[9] = 0x23;

            reenviar(OUTBuffer, 10);
        }
        try {
            if (periodo < 366) i = 12;
            else i = 10;
            enviar(OUTBuffer, i);
            Thread.Sleep(100);
            while (!recibir()) {
                enviar(OUTBuffer, i);
                Thread.Sleep(100);
            }

        }
        catch (Exception) {
            MessageBox.Show("Error en Sub()");
        }

    }

    public void Girar(long steps, int dir, uint periodo) {

        Byte[] OUTBuffer = new byte[65];
        timerConocer.Enabled = false;
        timerRecibir.Enabled = false;

        OUTBuffer[0] = 0x2D;
        OUTBuffer[1] = 0x92; //giro
        OUTBuffer[2] = Convert.ToByte(((steps & 0xFF0000) >> 16));
        OUTBuffer[3] = Convert.ToByte(((steps & 0xFF00) >> 8));
        OUTBuffer[4] = Convert.ToByte((steps & 0x00FF));
        OUTBuffer[5] = Convert.ToByte(dir);
        periodo /= 2;
        OUTBuffer[6] = Convert.ToByte(((periodo & 0xFF00) >> 8));
        OUTBuffer[7] = Convert.ToByte((periodo & 0x00FF));
        OUTBuffer[8] = checksum(OUTBuffer,2, 8);
        OUTBuffer[9] = 0x23;

        reenviar(OUTBuffer, 10);

        try {
            enviar(OUTBuffer, 10);
        }
        catch (Exception) {
            MessageBox.Show("Error en Girar()");
        }
    }

    public void Reset() {

        timerConocer.Enabled = false;
        timerRecibir.Enabled = false;

        Byte[] OUTBuffer = new byte[65];
        int periodox = 60;
        int periodoy = 160;

        OUTBuffer[0] = 0x2D;
        OUTBuffer[1] = 0x98;
        OUTBuffer[2] = Convert.ToByte((periodox & 0xFF00) >> 8);
```

```
        OUTBuffer[3] = Convert.ToByte(periodox & 0x00FF);
        OUTBuffer[4] = Convert.ToByte(((periodoy & 0xFF00) >> 8));
        OUTBuffer[5] = Convert.ToByte(periodoy & 0x00FF);
        OUTBuffer[6] = checksum(OUTBuffer,2, 6);
        OUTBuffer[7] = 0x23;

        reenviar(OUTBuffer, 8);

        try {
          enviar(OUTBuffer, 8);
          Is_Reset = true;
          timerReenviar.Enabled = true;
        }
        catch (Exception) {
          MessageBox.Show("Error en Reset()");
        }
      }

      #endregion



      /*Metodos que se puedan eliminar
       *
       */
      //Modificado con los metodos de conversion
      private string Calctiempototal() {
        try {
          int tiempototal = 0;
          //la primera posicion cogemos a partir de hpos
          tiempototal += Convert.ToInt32(Math.Abs(Convert.ToInt32(dataSet1.mitabla[0][2]) - Convert.ToInt32(hpos / avancepasgiro))*
10 / Convert.ToInt32(dataSet1.mitabla[0][3]));
          tiempototal += Convert.ToInt32(Math.Abs(Convert.ToInt32(dataSet1.mitabla[0][5]) - Convert.ToInt32(hpos / avancepasgiro))
* 10 / Convert.ToInt32(dataSet1.mitabla[0][7]));
          for (int i = 0; i < dataSet1.mitabla.Rows.Count; i++) {
            if (i != 0) {
              //pongo 0.7 seg de retraso por el giro
              tiempototal += 7 + Convert.ToInt32(Math.Abs(Convert.ToInt32(dataSet1.mitabla[i][2]) -
Convert.ToInt32(dataSet1.mitabla[i - 1][2])) * 10 / Convert.ToInt32(dataSet1.mitabla[i][3]));
            }
            int auxtdip = Convert.ToInt32(dataSet1.mitabla[i][4]);
            //pongo 0.7 seg de retraso por cada dipping
            tiempototal += auxtdip * 7 + 2 * auxtdip * Convert.ToInt32(Math.Abs(Convert.ToInt32(dataSet1.mitabla[i][6]) -
Convert.ToInt32(dataSet1.mitabla[i][5])) * 10 / Convert.ToInt32(dataSet1.mitabla[i][7]));
            //tiempototal += auxtdip * 7 + 2 * auxtdip * Convert.ToInt32(Math.Abs(Convert.ToInt32(dataSet1.mitabla[i][6]) -
Convert.ToInt32(dataSet1.mitabla[i][5])) * 10 / Convert.ToInt32(Convert.ToDouble(dataSet1.mitabla[i][7])));
            tiempototal += auxtdip * (Convert.ToInt32(dataSet1.mitabla[i][8]) + Convert.ToInt32(dataSet1.mitabla[i][9])) * 10;
          }
          tiempototal *= repeatotal;  //por el numero de repeticiones

          tiempototal = Convert.ToInt32(tiempototal / 10);
          int auxseg = 0, auxmin = 0, auxhoras = 0;
          auxseg = tiempototal % 60;
          auxmin = Convert.ToInt32(tiempototal / 60);
          auxhoras = Convert.ToInt32(auxmin / 60);
          auxmin = Convert.ToInt32(auxmin % 60);
          if (auxmin == 0 && auxhoras == 0) return auxseg + " seconds ";
          else if (auxhoras == 0) return auxmin + " minutes " + auxseg + " seconds";
          else return auxhoras + " hours " + auxmin + " minutes " + auxseg + " seconds";
        }
        catch (Exception) {
          return "ERROR al calcular el tiempo estimado";
        }
      }

      //Este no haría falta, ya que los compruebo yo antes de añadirlos a la tabla
      private bool RevisarDatos() {
        for (int i = 0; i < dataSet1.mitabla.Rows.Count; i++){ //Contador de filas del ciclo
          if ((int)dataSet1.mitabla[i][2] > 360) {
            MessageBox.Show("Maximum angle is 360º, Row: " + i);
            return false;
          }
```

```
            else if ((int)dataSet1.mitabla[i][3] > 180) {
              MessageBox.Show("Maximum rotation speed is 180º/s, Row: " + i);
              return false;
            }
            else if ((int)dataSet1.mitabla[i][3] <= 0) {
              MessageBox.Show("Wrong rotation speed, Row: " + i);
              return false;
            }
            else if ((double)this.dataSet1.mitabla[i][5] > 90) {
              MessageBox.Show("Maximum vale of H Up is 90 mm, Row: " + i);
              return false;
            }
            else if ((double)this.dataSet1.mitabla[i][6] > 90) {
              MessageBox.Show("Maximum vale of H Down is 90 mm, Row: " + i);
              return false;
            }
            else if ((double)this.dataSet1.mitabla[i][7] > 60) {
              MessageBox.Show("Maximum vertical speed is 60 mm/s, Row: " + i);
              return false;
            }
            else if ((double)this.dataSet1.mitabla[i][7] <= 0) {
              MessageBox.Show("Wrong rotation speed, Row: " + i);
              return false;
            }
          }
          return true;
        }

        //Este método es para borrar mas adelante junto con el boton en la pestaña Device.
        private void buttonClose_Click(object sender, EventArgs e) {
          this.Close();
        }
      }
    }
```

ANEXO 4: CÓDIGO PIC

```
#include <p18f4550.h>
#include <usart.h>
/** V A R I A B L E S ********************************************************/

#define SIZE                                    64
#define TRUE                                    1
#define FALSE                                   0

unsigned int Is_Busyx=0, Is_Busyy=0, resetx=0, resety=0,  Is_Reset=0, origenx = 1, origeny = 1, pasosreset = 0,  posunknown =
2;//origeny = 1,
unsigned int H_Dirx=1, H_Diry=1, num_div = 0, resto = 0, divisiones = 0;
unsigned int salida = 0, periodox = 400, periodoy = 200, sumacheck = 0, periodof = 400, stopmov = 0;
unsigned long contimer0=0, contimer1 = 2, distanciax = 0, distanciay = 0, num_pasos = 0, num_pasosy = 0;
unsigned int limitex = 59137, limitey = 12798, auxcntp = 0, auxnum_pasos = 0, auxdivisiones = 0, auxresto = 0, auxpasos = 0; //limite de
pasos de los dos ejes: 90*657.08 y 360*35.55

unsigned long aux_pasosy, aux_periodoy;
unsigned int aux_diry;

#pragma code
#define mInitAllLEDs()     LATD &= 0xF0; TRISD &= 0xF0
//                                                                                  las 4 lsb salidas=0; 4 lsb salidas digitale
#define mLED_1          LATDbits.LATD0
#define mLED_2          LATDbits.LATD1
#define mLED_3          LATDbits.LATD2
#define mLED_4          LATDbits.LATD3

#define mLED_1_On()       mLED_1 = 1
#define mLED_2_On()       mLED_2 = 1
#define mLED_3_On()       mLED_3 = 1
#define mLED_4_On()       mLED_4 = 1

#define mLED_1_Off()      mLED_1 = 0
#define mLED_2_Off()      mLED_2 = 0
#define mLED_3_Off()      mLED_3 = 0
#define mLED_4_Off()      mLED_4 = 0

#define mLED_1_Toggle()    mLED_1 = !mLED_1
#define mLED_2_Toggle()    mLED_2 = !mLED_2
#define mLED_3_Toggle()    mLED_3 = !mLED_3
#define mLED_4_Toggle()    mLED_4 = !mLED_4

/** S W I T C H ******************************************************/
#define mInitAllSwitches()  TRISBbits.TRISB4=1;TRISBbits.TRISB5=1
#define mInitSwitch2()     TRISBbits.TRISB4=1
#define mInitSwitch3()     TRISBbits.TRISB5=1
#define sw2             PORTBbits.RB4
#define sw3             PORTBbits.RB5



void CleanToSendDataBuffer(void);
void CleanReceivedDataBuffer(void);
void enviar(unsigned char out[SIZE], int j);
void enviarOK();
unsigned char checksum(unsigned char c[], int a, int b);

void ProcessIO(void);
//void InterruptHandlerHigh (void);
void YourHighPriorityISRCode(void);

#pragma udata
unsigned char inxbee = 0, h, l; //H y L para el checksum recibido
unsigned char ReceivedDataBuffer[SIZE], ToSendDataBuffer[SIZE];
int DataBufferL = -1,DataBufferT = SIZE, DataBufferS = 0, i;
```

```
#pragma code REMAPPED_HIGH_INTERRUP_VECTOR = REMAPPED_HIGH_INTERRUPT_VECTOR_ADRESS
void Remapped_High_ISR (void){
	_asm goto YourHighPriorityISRCode _endasm
}

//#pragma interrupt InterruptHandlerHigh
//void InterruptHandlerHigh (void)
#pragma interrupt YourHighPriorityISRCode
void YourHighPriorityISRCode()
{
	if (PIR1bits.RCIF) //Recepción por RS232
	{	//Interrupción por recepción
		//mLED_3_Toggle();
		PIR1bits.RCIF=0;

		//ReceivedDataBuffer[DataBufferL] = RCREG;
		//DataBufferL++;

		inxbee = RCREG;
		if ((DataBufferL != -1) && (DataBufferL != SIZE))
		{	//Una vez recivido el caracter lo pone en el buffer.
			//if (DataBufferL == SIZE) ReceivedDataBuffer[DataBufferT] = inxbee;
			//else
			ReceivedDataBuffer[DataBufferL] = inxbee;
			DataBufferL = DataBufferL + 1;
		}
		if ((inxbee==0x2D) && (DataBufferL == -1)) //'-'
		{  //Primer caracter de la cadena
			DataBufferL = 0;
		}
		else if ((inxbee==0x23) && (DataBufferL != -1))  //'#'
		{  //Fin de cadena en el caracter
			DataBufferT = DataBufferL;
			DataBufferL = SIZE;
		}
	}//Fin de PIR1bits.RCIF

	/*else if(PIR1bits.TXIF){ //Interrupcion por envio de datos. He encontrado la funcion putsUSART(char []);
		//mLED_1_Toggle();
		if (DataBufferS != Out)
		{
			TXREG = ToSendDataBuffer[Out];
			//WriteUSART(ToSendDataBuffer[Out]);
			Out++;
			//PIE1bits.TXIE = 0;
		}
		else {
			PIE1bits.TXIE = 0;
			DataBufferS = 0;
			Out = 0;
		}

			//for(Out=0; ToSendDataBuffer[Out] != 0x23; Out++){
				//WriteUSART(ToSendDataBuffer[Out]);
				//PIE1bits.TXIE=0;
			//}
			//WriteUSART(ToSendDataBuffer[Out]); //el caracter #
		//}
	}//Fin de PIR1bits.TXIF*/


	if (INTCONbits.TMR0IF)		//Interrupcion del timer 0 (temporizador) del numero de pasos del eje X
	{
		//mLED_1_Toggle();
		INTCONbits.TMR0IF=FALSE;			//Se desactiva el flag de la interrupcion


		if (PORTBbits.RB0 == 1 || origenx == 0) //Aqui no va a entrar, ya que RB0 esta siempre al aire = 0;
		{
			if (resetx == 1 && resety == 1)
			{
```

```
                        if (H_Dirx == 1)
                        {
                                contimer0 = 0;                  //cambiar para que no se haga
                                num_pasos = 0;
                                H_Dirx = 0;
                                PORTCbits.RC1= H_Dirx;//RC1
                                origenx = 0;
                        }
                        if (H_Dirx == 0 && origenx == 0 )
                        {
                                //mLED_1_Toggle();
                                PORTBbits.RB3 = !PORTBbits.RB3;         //forma de onda pasos para el eje x

                                if (periodox < 366) periodox = 366;         //velocidad maxima en el reseteo
                                TMR0H = 0xFF - (((periodox*6)&0xFF00)>>8);
                                TMR0L = 0xFF - ((periodox*6)&0x00FF);
                        }
                        if (PORTBbits.RB0 == 0 || pasosreset != 0)
                        {
                                if (pasosreset < 657)//recorre 1 mm
                                {
                                        pasosreset++;
                                        //mLED_1_Toggle();
                                        PORTBbits.RB3 = !PORTBbits.RB3;         //forma de onda pasos
para el eje x
                                        if (periodox < 366) periodox = 366;         //velocidad maxima en el
reseteo
                                        TMR0H = 0xFF - (((periodox*6)&0xFF00)>>8);
                                        TMR0L = 0xFF - ((periodox*6)&0x00FF);
                                }
                                else
                                {
                                        Is_Busyx = 0;
                                        Is_Busyy= 0;
                                        origenx = 1;
                                        resetx = 0;
                                        distanciax = 0;//-= num_pasos;
                                        distanciay = 0;
                                        if (Is_Reset > 0) Is_Reset = 0; //cuando llegue a cero sera reset
completado
                                        if (posunknown > 0) posunknown--;
                                        contimer0 = 0;
                                        T0CONbits.TMR0ON = 0;
                                        PORTBbits.RB3=0;
                                        num_div = 0;                                    //Se inicializa
la variable para el siguiente proceso
                                        resto = 0;
                                        divisiones = 0;
                                        num_pasos = 0;
                                        resety = 0;
                                        pasosreset = 0;

                                }
                        }
                }
                else if ( resetx == 0 )
                {
                        if (pasosreset == 0)    //desactivar el CCP2 para resetear posicion solo con el timer0
                        {
                                periodox = 366;
                                CCP2CON = 0x00;
                                T0CON = 0x88;
                        }
                        if ( pasosreset < 2631 )            //recorre 4 mm para que el detector lea siempre 1 y no
varie
                        {
                                pasosreset++;
                                //mLED_1_Toggle();
                                PORTBbits.RB3 = !PORTBbits.RB3; //RB3     //forma de onda pasos para el eje x

                                if (periodox < 366) periodox = 366;         //velocidad maxima en el reseteo
                                TMR0H = 0xFF - (((periodox*6)&0xFF00)>>8);
```

```
                                        TMR0L = 0xFF - ((periodox*6)&0x00FF);

                                }
                                else
                                {
                                        resetx = 1;
                                        H_Dirx = 0;
                                        contimer0 = 0;
                                        PORTCbits.RC1= H_Dirx;
                                        origenx = 0;
                                        pasosreset = 0;
                                        distanciax = 0;
                                        distanciay = 0;

                                        resety = 1;
                                        origeny = 0;
                                        T0CONbits.TMR0ON = 0;
                                        //Temporizador 1
                                        T1CON = 0x81;
                                        PIE1bits.TMR1IE = 1;
                                        TMR1H = 0xFF - (((periodoy*6)&0xFF00)>>8);
                                        TMR1L = 0xFF - ((periodoy*6)&0x00FF);

                                        IPR1bits.TMR1IP = 1;                    //prioridad
                                }
                        }
        }
        else if (periodof < 366)
        {
                num_div++;

                if (H_Dirx == 0) distanciax += num_pasos;       //Se actualiza la distancia recorrida una vez
                else distanciax -= num_pasos;
                if (distanciax > limitex && H_Dirx == 1) distanciax = 0;
                else if (distanciax > limitex && H_Dirx == 0) distanciax = limitex;
                //PORTCbits.RC0 = !PORTCbits.RC0;
                if (stopmov == 1 && (num_div != divisiones) && Is_Reset == 0 )
                {
                        stopmov = 0;
                        INTCON2bits.TMR0IP = 1;         //Prioridad alta para la interrupcion del timer 0 y baja para
el timer 1 y CCP
                        CCP2CON = 0x00;                         //Se desactiva al CCP2
                        T0CONbits.TMR0ON = 0;           // Se desactiva el timer 0
                        INTCON2bits.TMR0IP = 0;         //Una vez hecha la operación vuelve a ser prioridad baja

                        num_div = 0;                            //Se inicializa la variable para el siguiente
proceso
                        resto = 0;
                        divisiones = 0;
                        num_pasos = 0;
                        Is_Busyx = 0;
                }
                else if ((num_div + 1) == divisiones && (resto != 0))   //En la ultima division se suma el resto
                {
                        TMR0H = 0xFF - (((num_pasos+resto)&0xFF00)>>8);
                        TMR0L = 0xFF - ((num_pasos+resto)&0x00FF);
                }
                else if (num_div == divisiones && num_pasos != 0)       //cuando se iguala a la distancia el numero total de pasos -
> se para
                {
                        INTCON2bits.TMR0IP = 1;         //Prioridad alta para la interrupcion del timer 0 y baja para
el timer 1 y CCP
                        //T1CONbits.TMR1ON = 0;         //Se desactiva el timer 1
                        CCP2CON = 0x00;                         //Se desactiva al CCP2
                        T0CONbits.TMR0ON = 0;           // Se desactiva el timer 0
                        INTCON2bits.TMR0IP = 0;         //Una vez hecha la operación vuelve a ser prioridad baja

                        if (H_Dirx == 0) distanciax += resto;           //Se actualiza la distancia recorrida una vez
                        else distanciax -= resto;
                        num_div = 0;                            //Se inicializa la variable para el siguiente
proceso
                        resto = 0;
```

```
                        divisiones = 0;
                        num_pasos = 0;
                        Is_Busyx = 0;
                        stopmov = 0;
                }
                else
                {
                        TMR0H = 0xFF - (((num_pasos)&0xFF00)>>8);
                        TMR0L = 0xFF - ((num_pasos)&0x00FF);
                }

        }
        else
        {
                        //mLED_1_Toggle();
                        PORTBbits.RB3 = !PORTBbits.RB3;         //forma de onda pasos para el eje x
                        contimer0++;
                        //T0CON = 0x88;
                        TMR0H = 0xFF - (((periodox*6)&0xFF00)>>8);
                        TMR0L = 0xFF - ((periodox*6)&0x00FF);

                        if ( contimer0 % 2 == 0 && num_pasos != 0 ) //Si esta haciendo reset (resetx=0) no entra
                        {
                                if (stopmov == 1 && contimer0 < num_pasos*2)
                                {
                                        stopmov = 0;
                                        T0CONbits.TMR0ON = 0;                   // Se desactiva el timer 0
                                        if (posunknown == 0)
                                        {
                                                if (H_Dirx == 0) distanciax += contimer0 / 2;           //Se actualiza
la distancia recorrida una vez
                                                else distanciax -= contimer0 / 2;
                                                if (distanciax > limitex && H_Dirx == 1) distanciax = 0;
                                                else if (distanciax > limitex && H_Dirx == 0) distanciax = limitex;
                                        }
                                        PORTBbits.RB3=0; //cuando acaba el movimiento se pone a cero para el
siguiente movimiento
                                        contimer0 = 0;                  //cambiar para que no se haga
                                        num_pasos = 0;
                                        Is_Busyx = 0;
                                }

                                else if ( contimer0 >= num_pasos*2  )   //cuando se iguala al numero de pasos -> se
para
                                {
                                        Is_Busyx = 0;
                                        T0CON = 0x00;
                                        T0CONbits.TMR0ON = 0;                   // Se desactiva el timer 0
                                        if (posunknown == 0)
                                        {
                                                if (H_Dirx == 0) distanciax += num_pasos;               //Se actualiza
la distancia recorrida una vez
                                                else distanciax -= num_pasos;
                                                if (distanciax > limitex && H_Dirx == 1) distanciax = 0;
                                                else if (distanciax > limitex && H_Dirx == 0) distanciax = limitex;
                                        }
                                        PORTBbits.RB3=0; //cuando acaba el movimiento se pone a cero para el
siguiente movimiento
                                        contimer0 = 0;                  //cambiar para que no se haga
                                        num_pasos = 0;
                                        stopmov = 0;
                                }
                        }
                }
        }//Fin deltimer 0


        if (PIR1bits.TMR1IF)   //Interrupcion del timer 1 (temporizador) del numero de pasos del eje Y
        {
                PIR1bits.TMR1IF = 0;
                //PORTCbits.RC0 = !PORTCbits.RC0;
```

```
		if (PORTBbits.RB1 == 0 && origeny == 0) //|| origeny == 0)
		{
			resety = 1;
			origeny = 1;

			//añadido
			distanciay = 0;
			//if (Is_Reset > 0) Is_Reset--; //cuando llegue a cero sera reset completado
			if (posunknown > 0) posunknown--;
			contimer1 = 2;
			T1CONbits.TMR1ON = 0;
			PORTAbits.RA3=0;
			T0CON = 0x88;
			TMR0H = 0xFF - (((periodox*6)&0xFF00)>>8);
			TMR0L = 0xFF - ((periodox*6)&0x00FF);
		}

		else
		{
			//mLED_2_Toggle();
			PORTAbits.RA3 = !PORTAbits.RA3;		//forma de onda pasos para el eje x

			//T1CON = 0x81;
			TMR1H = 0xFF - (((periodoy*6)&0xFF00)>>8);
			TMR1L = 0xFF - ((periodoy*6)&0x00FF);

			if ( contimer1 % 2 == 0 && num_pasosy != 0 && posunknown == 0 ) //Si esta haciendo reset (resetx=0)
no entra
			{
				if (H_Diry == 0)
				{
					distanciay++;
					if (distanciay >= limitey)
					{
						distanciay = limitey;
						stopmov = 0;
						T1CONbits.TMR1ON = 0;
						PORTAbits.RA3=0;	//cuando acaba el movimiento se pone a cero
para el siguiente movimiento
						contimer1 = 1;			//cambiar para que no se haga
						Is_Busyy = 0;
						num_pasosy = 0;
					}
				}
				else
				{
					if ( distanciay > 0 ) distanciay--;
					else //(distanciay <= 0)
					{
						distanciay = 0;
						stopmov = 0;
						T1CONbits.TMR1ON = 0;
						PORTAbits.RA3=0;	//cuando acaba el movimiento se pone a cero
para el siguiente movimiento
						contimer1 = 1;			//cambiar para que no se haga
						Is_Busyy = 0;
						num_pasosy = 0;
					}
				}
				if ((stopmov == 1 && contimer1 < num_pasosy*2) || contimer1 >= num_pasosy*2)
				{
					stopmov = 0;
					T1CONbits.TMR1ON = 0;		// Se desactiva el timer 1
					PORTAbits.RA3=0;		//cuando acaba el movimiento se pone a cero
para el siguiente movimiento
					contimer1 = 1;				//cambiar para que no se haga
					Is_Busyy = 0;
					num_pasosy = 0;
				}
			}
		contimer1++;
		}
```

```
		}//Fin del PIR1bits.TMR1IF

} //Fin de void YourHighPriorityISRCode();


void main(void)
{
		//Configuración del pic

		mInitAllLEDs();

		ADCON1 = 0x0F;
		TRISDbits.TRISD3 = 0;
		TRISCbits.TRISC7=1; // RX
  TRISCbits.TRISC6=0; // TX

		TRISBbits.TRISB3 = 0; //Salida
		TRISCbits.TRISC1 = 0;
		TRISAbits.TRISA0 = 0;

		TRISAbits.TRISA3 = 0; //Salida
		TRISCbits.TRISC2 = 0;
		TRISAbits.TRISA5 = 0;

		//Para la configuración del RS232
		TXSTA=0x24;			// Inicialización transmisión
		RCSTA = 0x90;	// continuous RX

		BAUDCON = 0x08;

		SPBRG = 0xD0;			// 0x04E2 -> 9600 baud
  SPBRGH = 0x00;	// 0x0271 -> 19200 baud
								// 0x0138 -> 38400 baud
								// 0x00D6 -> 56000 baud
								// 0x00D0 -> 57600 baud
								// 0x0068 -> 115200 baud //Solo recibe 4 bytes y se para


		RCONbits.IPEN=0;	// Sistema de prioridad en las interrupciones deshabilitado
		INTCONbits.GIEH=1;	// Interrupciones habilitadas
		INTCONbits.GIEL=1;	// Interrupciones de perifericos habilitadas
		PIE1bits.RCIE=1;	// Habilita la recepción EUSART
		PIE1bits.TXIE=0;	// Habilita la transmision EUSART

		//Empieza el programa
		CleanReceivedDataBuffer();
		CleanToSendDataBuffer();
		while (1)
  {
			if (DataBufferL == SIZE)
			{
				mLED_4_Toggle();
				ProcessIO();
			}
			//if (RCSTAbits.OERR) {
			//	mLED_4_On();
			//	RCSTA = 0x00;
			//	RCSTA = 0x90;
			//}
			//WriteUSART(0xAA);
  }
}

void CleanReceivedDataBuffer(void)
{
		for (i = 0; i<=DataBufferT; i++) ReceivedDataBuffer[i] = FALSE;
}

void CleanToSendDataBuffer(void) {
		for (i = 0; i!=SIZE; i++) ToSendDataBuffer[i] = FALSE;
}
```

```
unsigned char checksum(unsigned char c[], int a, int b){
	sumacheck = 0;
	for (i = a; i < b; i++) sumacheck += c[i];
	return (sumacheck)&0x00FF;
}

void enviarOK(){
	ToSendDataBuffer[0] = 0x2D; //'-'
	ToSendDataBuffer[1] = 0x4F;	//'O'
	ToSendDataBuffer[2] = 0x6B;	//'k'
	ToSendDataBuffer[3] = 0x23;	//'#'
	mLED_3_Toggle();
	enviar(ToSendDataBuffer,4);
}

void enviar(unsigned char out[SIZE], int j){
	i = 0;
	while (i < j) {
		if (PIR1bits.TXIF) { //Comprobamos si el TXREG esta vacio.
			TXREG = out[i];
			i++;
		}
	}
}

void ProcessIO()
{

	switch (ReceivedDataBuffer[0])
	{
		case 0x80:
			mLED_1_Toggle();
			mLED_2_Toggle();
		break; //Fin del 0x80

		//Envio
		case 0x81:

			mLED_1_Toggle();
			ToSendDataBuffer[0] = 0x2d;
			ToSendDataBuffer[1] = 0x81;
			ToSendDataBuffer[2] = posunknown;
			ToSendDataBuffer[3] = Is_Busyx;
			ToSendDataBuffer[4] = Is_Busyy;
			ToSendDataBuffer[5] = Is_Reset;
			ToSendDataBuffer[6] = checksum(ToSendDataBuffer,2,6);
			ToSendDataBuffer[7] = 0x23;
			enviar(ToSendDataBuffer,8);	//polling

			//Por interrupción
			//Out = 1;				//Primer byte a enviar cuando se produzca la interrupcion
			//DataBufferS = 9;	//maximo de bytes a enviar mas 1.
			//TXREG = ToSendDataBuffer[0];		//Enviamos el primer byte
			//PIE1bits.TXIE = 1;					//Activamos la interrupción

		break; //Fin del 0x81

		//Envio
		case 0x37:

			mLED_2_Toggle();
			ToSendDataBuffer[0] = 0x2d;
			ToSendDataBuffer[1] = 0x37;
			if (periodox < 366 && H_Dirx == 1)
			{
				ToSendDataBuffer[2] =  (distanciax)&0x00FF;		//(distanciax)&0x00FF; ((distanciax +
((contimer0) / 2)*(-2*H_Dirx+1); + (num_div * num_pasos / 2))*
				ToSendDataBuffer[3] =  ((distanciax)&0xFF00)>>8;// ((distanciax)&0xFF00)>>8;
				ToSendDataBuffer[4] =  ((distanciax)&0xFF0000)>>16; //((distanciax)&0xFF0000)>>16;

				if (distanciax <= 0 && Is_Reset == 0)
				{
```

```
                                        distanciax = 0;
                                        CCP2CON = 0x00;                         //Se desactiva al CCP2
                                        T0CONbits.TMR0ON = 0;                   // Se desactiva el timer 0
                                        INTCON2bits.TMR0IP = 0;                 //Una vez hecha la operación vuelve
a ser prioridad baja
                                        num_div = 0;                            //Se inicializa la variable
para el siguiente proceso
                                        resto = 0;
                                        divisiones = 0;
                                        num_pasos = 0;
                                        Is_Busyx = 0;
                                }
                        }
                        else if (periodox < 366 && H_Dirx == 0)
                        {
                                ToSendDataBuffer[2] =  (distanciax)&0x00FF;
                                ToSendDataBuffer[3] =  ((distanciax)&0xFF00)>>8;
                                ToSendDataBuffer[4] =  ((distanciax)&0xFF0000)>>16;
                                if (distanciax >= limitex && Is_Reset == 0)
                                {
                                        //distanciax = 0;
                                        CCP2CON = 0x00;                         //Se desactiva al CCP2
                                        T0CONbits.TMR0ON = 0;                   // Se desactiva el timer 0
                                        INTCON2bits.TMR0IP = 0;                 //Una vez hecha la operación vuelve
a ser prioridad baja
                                        num_div = 0;                            //Se inicializa la variable
para el siguiente proceso
                                        distanciax = limitex;
                                        resto = 0;
                                        divisiones = 0;
                                        num_pasos = 0;
                                        Is_Busyx = 0;

                                }
                        }
                        else if (periodox > 366 && H_Dirx == 0)
                        {
                                ToSendDataBuffer[2] =  (distanciax + ((contimer0) / 2))&0x00FF;  //(distanciax)&0x00FF;
((distanciax + ((contimer0) / 2)*(-2*H_Dirx+1); + (num_div * num_pasos / 2))*
                                ToSendDataBuffer[3] =  ((distanciax + ((contimer0) / 2))&0xFF00)>>8;//
((distanciax)&0xFF00)>>8;
                                ToSendDataBuffer[4] =  ((distanciax + ((contimer0) / 2))&0xFF0000)>>16;
//((distanciax)&0xFF0000)>>16;
                                if (distanciax + (contimer0 / 2) > limitex && Is_Reset == 0) //stopmov = 1;
                                {
                                        distanciax = limitex;
                                        T0CONbits.TMR0ON = 0;                   // Se desactiva el timer 0

                                        PORTBbits.RB3=0; //cuando acaba el movimiento se pone a cero para el
siguiente movimiento
                                        contimer0 = 0;                  //cambiar para que no se haga
                                        num_pasos = 0;
                                        Is_Busyx = 0;

                                }
                        }
                        else
                        {
                                ToSendDataBuffer[2] =  (distanciax - ((contimer0) / 2))&0x00FF;  //(distanciax)&0x00FF;
((distanciax + ((contimer0) / 2)*(-2*H_Dirx+1); + (num_div * num_pasos / 2))*
                                ToSendDataBuffer[3] =  ((distanciax - ((contimer0) / 2))&0xFF00)>>8;//
((distanciax)&0xFF00)>>8;
                                ToSendDataBuffer[4] =  ((distanciax - ((contimer0) / 2))&0xFF0000)>>16;
//((distanciax)&0xFF0000)>>16;
                                if (distanciax - (contimer0 / 2) < 0 && Is_Reset == 0) //stopmov = 1;
                                {
                                        distanciax = 0;
                                        T0CONbits.TMR0ON = 0;                   // Se desactiva el timer 0

                                        PORTBbits.RB3=0; //cuando acaba el movimiento se pone a cero para el
siguiente movimiento
                                        contimer0 = 0;                  //cambiar para que no se haga
```

```
                                        num_pasos = 0;
                                        Is_Busyx = 0;


                                }
                        }

                        ToSendDataBuffer[5] = (distanciay)&0x00FF;//(distanciay + (((contimer1 - 2) / 2)))&0x00FF;
                        ToSendDataBuffer[6] = ((distanciay)&0xFF00)>>8;//((distanciay + ((contimer1 - 2) / 2))&0xFF00)>>8;
                        ToSendDataBuffer[7] = ((distanciay)&0xFF0000)>>16;//((distanciay + ((contimer1 - 2) /
2))&0xFF0000)>>16;
                        ToSendDataBuffer[8] = checksum(ToSendDataBuffer,2,8);
                        ToSendDataBuffer[9] = 0x23;

                        //Polling TX
                        enviar(ToSendDataBuffer,10);

                        //Interrupción
                        //Out = 1;                      //Primer byte a enviar cuando se produzca la interrupcion
                        //DataBufferS = 11;   //maximo de bytes a enviar mas 1.

                        //TXREG = ToSendDataBuffer[0];          //Enviamos el primer byte
                        //PIE1bits.TXIE = 1;                            //Activamos la interrupción

                break; //Fin del 0x37

                //Recepción
                case 0x85:              //solo para movimiento del eje altura para periodo menor de 60 microsegundos

                        if (checksum(ReceivedDataBuffer,1,9) == ReceivedDataBuffer[9])
                        { //Habra que mover todo esto uno para dentro
                                enviarOK();
                                if (Is_Reset == 0)
                                {
                        //Is_Busyx = 1;         //Ocupado hasta que no pare de hacer la operación
                                        auxdivisiones = (int)ReceivedDataBuffer[1];
                                        auxnum_pasos =(int)ReceivedDataBuffer[3] + (int)ReceivedDataBuffer[2]*0x100
+ 0*0x10000 + 0*0x1000000;
                                        auxresto = (int)ReceivedDataBuffer[4];      //Se recibe el numero de divisiones,
el valor de la division y el resto
                                        H_Dirx=(int)ReceivedDataBuffer[5];          //Direccion

                                        periodox = (long)ReceivedDataBuffer[8] + (long)ReceivedDataBuffer[7]*0x100;
                                        //(long)ReceivedDataBuffer[6]*0x10000+

                                        periodof = periodox;
                                        PORTCbits.RC1 = H_Dirx;
                                        PORTAbits.RA0 = 1;   //enable para el eje x

                                        if (H_Dirx == 0 && (distanciax + num_div * auxnum_pasos + resto) > limitex)
                                        {
                                                auxcntp = (limitex - distanciax) / 0xFF;//FFF;
                                                //((num_div * num_pasos + resto) - (limite - distanciax)) / num_pasos
                                                if ((limitex - distanciax)% 0xFF != 0) auxcntp++;//FFF
                                        divisiones = (limitex - distanciax) / auxcntp;
                                        resto = (limitex - distanciax) % auxcntp;

                                        }
                                        else if (H_Dirx == 1 && (distanciax - (num_div * auxnum_pasos + resto)) < 0)
                                        {
                                                auxcntp = distanciax / 0xFF;//FFF

                                                if (distanciax % 0xFF != 0) auxcntp++;//FFF
                                divisiones = distanciax / auxcntp;
                        resto = distanciax % auxcntp;
                                        }
                                        if (Is_Busyx == 0)
                                        {
                                                Is_Busyx = 1;
                                                num_pasos = auxnum_pasos;
                                                divisiones = auxdivisiones;
                                                resto = auxresto;
```

```
                                        if (H_Dirx == 0 && (distanciax + divisiones * num_pasos + resto) >
limitex)
                                        {
                                                auxcntp = (limitex - distanciax) / 0xFF;//FFF
                                                if ((limitex - distanciax)% 0xFF != 0) auxcntp++; //FFF
                                divisiones = (limitex - distanciax) / auxcntp;
                                resto = (limitex - distanciax) % auxcntp;

                                        }
                                        else if (H_Dirx == 1 && (distanciax - (divisiones * auxnum_pasos +
resto)) < 0)
                                        {
                                                auxcntp = distanciax / 0xFF; //FFF

                                                if (distanciax % 0xFF != 0) auxcntp++;//FFF
                                divisiones = distanciax / auxcntp;
                                resto = distanciax % auxcntp;
                                        }
                                        if (periodox <= 21)T2CON = 0x04;                //pre-escalar 1:1
                                        else if (periodox > 21 && periodox <= 84)
                                        {
                                                T2CON = 0x05; //pre-escalar 1:4
                                                periodox = periodox / 4;                //Se divide entre 4
porque utilizamos pre-escalar 4
                                        }
                                        else
                                        {
                                                T2CON = 0x07;  //Con pre-escalar 16
                                                periodox = periodox / 16;
                                        }

                                        CCP2CON = 0x0F;       //Modo ccp para PWM
                                        PR2 = periodox * 12;  // 48 MHz reloj / 4
                                        //CTon = PR2*2 (Duty 50%)
                                        CCPR2L = PR2 / 2;
                                        CCP2CONbits.DC2B0=(periodox*24)&0x0001;
                                        CCP2CONbits.DC2B1=(periodox*24)&0x0002;
                                        //Temporizador 0
                                        T0CON = 0xA8;                           //Cuenta cada pulso recibido en A4,
sin pre-escalar
                                        INTCONbits.TMR0IE = 1;


                                        TMR0H = 0xFF - (((num_pasos)&0xFF00)>>8);
                                        TMR0L = 0xFF - ((num_pasos)&0x00FF);
                                        INTCON2bits.TMR0IP = 1;
                                }
                                else
                                {
                                        if (H_Dirx == 0 && ((distanciax + (divisiones - num_div) * num_pasos
+ resto)+ auxdivisiones * auxnum_pasos + auxresto) > limitex)
                                        {
                                                auxcntp = (limitex - distanciax) / num_pasos;
                                                //((num_div * num_pasos + resto) - (limite - distanciax)) /
num_pasos

                                                if ((limitex - distanciax)% num_pasos != 0 && resto == 0)
auxcntp++;
                                        divisiones += auxcntp;
                                        resto += (limitex - distanciax + num_div * num_pasos)% num_pasos;
                                        }
                                        else
                                        {
                                                //auxpasos = auxnum_pasos * auxdivisiones + auxresto;
                                                auxcntp = (auxdivisiones * auxnum_pasos + auxresto) /
num_pasos;

                                                if ((auxdivisiones * auxnum_pasos + resto)% num_pasos !=
0 && resto == 0) auxcntp++;
                                        divisiones += auxcntp;
                                        resto += (auxdivisiones * auxnum_pasos + auxresto)% num_pasos;
                                        }
                                }
                        }
```

```
                        } //Fin del if h && l //Comprobación del checksum

    break; //Case 0x85

                //Recepción
                case 0x90: //solo para movimiento del eje altura para periodo mayor de 60 microsegundos


                        if (checksum(ReceivedDataBuffer,1,7) == ReceivedDataBuffer[7])
                        { //Habra que mover todo esto uno para dentro
                                enviarOK();
                                if (Is_Reset == 0)
                                {
                                        Is_Busyx = 1;           //Ocupado hasta que no pare de hacer la operación

                                        H_Dirx=(int)ReceivedDataBuffer[4];          //Direccion
                                        PORTCbits.RC1 = H_Dirx;//RC1
                                        periodox = (long)ReceivedDataBuffer[6] + (long)ReceivedDataBuffer[5]*0x100;

                                        PORTAbits.RA0 = 1;   //enable para el eje x

                                        periodof = periodox;

                                        if (num_pasos != 0)
                                        {
                                                num_pasos +=(long)ReceivedDataBuffer[3] +
(long)ReceivedDataBuffer[2]*0x100 + (long)ReceivedDataBuffer[1]*0x10000 + 0*0x1000000;
                                                if (H_Dirx == 0 && distanciax + num_pasos > limitex) num_pasos =
limitex - distanciax;
                                                else if (H_Dirx == 1 && distanciax - num_pasos < 0) num_pasos =
distanciax;
                                        }
                                        else
                                        {

                                                num_pasos =(long)ReceivedDataBuffer[3] +
(long)ReceivedDataBuffer[2]*0x100 + (long)ReceivedDataBuffer[1]*0x10000 + 0*0x1000000;

                                                if (H_Dirx == 0 && distanciax + num_pasos > limitex) num_pasos =
limitex - distanciax;
                                                else if (H_Dirx == 1 && distanciax - num_pasos < 0) num_pasos =
distanciax;
                                                //Temporizador 0
                                                if (periodox < 5460)   //No hace falta preesclar (hasta 3,4 mm/s en el
robot)
                                                {
                                                        T0CON = 0x88;                   // Cuando llega el
temporizador al periodo se activa el flag, sin pre-escalar
                                                }
                                                else if (periodox < 349440)       //Preescalar 64 se cambia el T0CON
y se asigna el nuevo periodo
                                                {
                                                        T0CON = 0x85;
                                                        periodox = periodox / 64;
                                                }
                                                else        //Preescalar 256 se cambia el T0CON y se asigna el nuevo
periodo
                                                {
                                                        T0CON = 0x87;
                                                        periodox = periodox / 256;
                                                }
                                                //Temporizador 0
                                                //T0CON = 0xA8;                 //Cuenta cada pulso recibido en A4,
sin pre-escalar
                                                INTCONbits.TMR0IE = 1;
                                                TMR0H = 0xFF - (((periodox*6)&0xFF00)>>8);
                                                TMR0L = 0xFF - ((periodox*6)&0x00FF);

                                                INTCON2bits.TMR0IP = 1;
                                        }
                                }
                        }//fin del if H y L
```

```
                    break; //Case 0x90

                    //Recepción
                    case 0x92: //solo para movimiento del eje giro

                              if (checksum(ReceivedDataBuffer,1,7) == ReceivedDataBuffer[7])
                              { //Habra que mover todo esto uno para dentro
                                        enviarOK();
                                        if (Is_Reset == 0)
                                        {
                                        //Is_Busyy = 1;
//Ocupado hasta que no pare de hacer la operación
                                                  H_Diry =(int)ReceivedDataBuffer[4];          //Direccion
                                                  PORTCbits.RC2 = H_Diry;
                                                  periodoy = (long)ReceivedDataBuffer[6] + (long)ReceivedDataBuffer[5]*0x100;
                                                  PORTAbits.RA5 = 1;

                                                  if (Is_Busyy == 1)//num_pasosy != 0)
                                                  {
                                                            num_pasosy += (long)ReceivedDataBuffer[3] +
(long)ReceivedDataBuffer[2]*0x100 + (long)ReceivedDataBuffer[1]*0x10000 + 0*0x1000000;
                                                            if (H_Diry == 0 && distanciay + num_pasosy > limitey)
                                                            {
                                                                      num_pasosy = limitey - distanciay;
                                                            }
                                                            else if (H_Diry == 1 && distanciay - num_pasosy < 0)
                                                            {
                                                                      num_pasosy = distanciay;
                                                            }
                                                  }
                                                  else
                                                  {
                                                            Is_Busyy = 1;
                                                            num_pasosy = (long)ReceivedDataBuffer[3] +
(long)ReceivedDataBuffer[2]*0x100 + (long)ReceivedDataBuffer[1]*0x10000 + 0*0x1000000;

                                                            if (H_Diry == 0 && distanciay + num_pasosy > limitey) num_pasosy =
limitey - distanciay;
                                                            else if (H_Diry == 1 && distanciay - num_pasosy < 0) num_pasosy =
distanciay;
                                                            //Temporizador 1
                                                            if (periodoy < 5460)
//No hace falta preesclar
                                                            {
                                                                      T1CON = 0x81;
// Cuando llega el temporizador al periodo se activa el flag, sin pre-escalar
                                                            }
                                                            else                    //Preescalar 8
                                                            {
                                                                      T1CON = 0xB1;
                                                                      periodoy = periodoy / 8;
                                                            }
                                                            //T1CON = 0x81;
                              // Cuando llega el temporizador al periodo se activa el flag, sin pre-escalar
                                                            PIE1bits.TMR1IE = 1;
                                                            TMR1H = 0xFF - (((periodoy*6)&0xFF00)>>8);
                                                            TMR1L = 0xFF - ((periodoy*6)&0x00FF);
                                                            IPR1bits.TMR1IP = 1;
                                                  }
                                        }
                              }//Fin del if h y l
                    break; //Case 0x92

                    //Recepción
                    case 0x96:
                              stopmov = 1;
                    break; //Case 0x96

                    //Recepción
                    case 0x98:
                              if (checksum(ReceivedDataBuffer,1,5) == ReceivedDataBuffer[5])
```

```
                                { //Habra que mover todo esto uno para dentro
                                        enviarOK();
                                        Is_Reset = 1;
                                        num_pasos = 0; num_pasosy = 0;
                                        //periodox = 100; periodoy = 100; //para el reset siempre velocidad 500
                                        periodox = (long)ReceivedDataBuffer[2] + (long)ReceivedDataBuffer[1]*0x100;
                                        periodoy = (long)ReceivedDataBuffer[4] + (long)ReceivedDataBuffer[3]*0x100;
                                        H_Dirx =(int)ReceivedDataBuffer[5];
                                        H_Diry =(int)ReceivedDataBuffer[6];
                                        PORTCbits.RC1 = H_Dirx;//RC1 = 1;
                                        PORTCbits.RC2 = H_Diry;

                                        T1CONbits.TMR1ON = 0;
//Timer1 off para que no gire, solo suba
                                        if (PORTBbits.RB0 == 1) resetx == 1;

                                        //Temporizador 0
                                        /*T0CON = 0x88;
// Cuando llega el temporizador al periodo se activa el flag, sin pre-escalar
                                        INTCONbits.TMR0IE = 1;
                                        TMR0H = 0xFF - (((periodox*12)&0xFF00)>>8);
                                        TMR0L = 0xFF - ((periodox*12)&0x00FF);
                                        INTCON2bits.TMR0IP = 1;*/
                                        periodof = periodox;
                                        if (periodox <= 21)T2CON = 0x04;                //pre-escalar 1:1
                                        else if (periodox > 21 && periodox <= 84)
                                        {
                                                T2CON = 0x05; //pre-escalar 1:4
                                                periodox = periodox / 4;
                                                //Se divide entre 4 porque utilizamos pre-escalar 4
                                        }
                                        else
                                        {
                                                T2CON = 0x07;
                                                periodox = periodox / 16;
                                        }

                                        CCP2CON = 0x0F;       //Modo ccp para PWM
                                        PR2 = periodox * 12;//periodox * 12;          // 48 MHz reloj / 4
                                        //CTon = PR2*2 (Duty 50%)
                                        CCPR2L = PR2 / 2;
                                        CCP2CONbits.DC2B0=(periodox*24)&0x0001; //(periodox*24)&0x0001;
                                        CCP2CONbits.DC2B1=(periodox*24)&0x0002; //(periodox*24)&0x0002;
                                        //Temporizador 0
                                        T0CON = 0xA8;
                                        //Cuenta cada pulso recibido en A4, sin pre-escalar
                                        INTCONbits.TMR0IE = 1;
                                        num_pasos = 10;
                                        divisiones = 0;
                                        resto = 0;
                                        TMR0H = 0xFF - (((num_pasos)&0xFF00)>>8);
                                        TMR0L = 0xFF - ((num_pasos)&0x00FF);
                                        INTCON2bits.TMR0IP = 1;
                                } //Fin del h y l
                        break; //Case 0x98
                }//Fin del switch (ReceivedDataBuffer[0])
                CleanReceivedDataBuffer();
                DataBufferL = -1;
}
```